

# PC-8801

N88-BASIC

解析マニュアル

川村 清 著

SHUWA  
SYSTEM  
TRADING  
CO.,LTD.

## 御注意

- (1) 本書は著者らが独自に解析した結果を出版したものです。
- (2) 本書は内容について万全を期して作製いたしましたが、御不審な点や誤り、記載もれなどお気付きのことがありましたら、出版元まで書面にて御連絡ください。
- (3) 本書の内容に関して運用した結果の影響については(2)項にかかわらず責任を負いかねますので御了承ください。
- (4) 本書の全部または一部について出版元から文書による許諾を得ずに、いかなる方法においても無断で複写、複製することは禁じられています。

***PC-8801***

***N88-BASIC***

***ANALYTICAL MANUAL***

**SHUWA SYSTEM TRADING CO.,LTD.**





## ま え が き

日本電気のパーソナル・コンピュータPC-8801は、以前に発表され多くの実績を誇るPC-8001の上位機種として、1981年12月に発売されました。PC-8001と比較したPC-8801の特徴は、184Kバイトのメモリが標準実装されたこと、漢字ROMを装備する事により日本語処理が可能になったこと、内蔵のRS-232Cチャンネルに対するサポートが強化されたこと、グラフィック機能の大幅な強化がなされたこと、周辺に対する拡張性が向上したこと、などがあげられます。

これに伴って、N88-BASICインタプリタの内部においても、メモリ・バンクのスイッチング、高解像度グラフィック等を中心としたサポート・ルーチンやI/OCSが強化されています。

PC-8801は、8ビットのCPUを用いた究極のパーソナル・コンピュータとも云われている反面、多機能化を実現するためにBASICプログラムの相対的な実行速度の低下、機械語によるプログラミングの複雑さなどの弱点もいくつか見受けられます。

本書はPC-8801のユーザが上記の優れた機能を十分に引き出し、PC-8801の持てる機能を思うままに活用できるよう、筆者らが独自に解析したN88-BASICインタプリタの内容をまとめあげ、ここに発表するものであります。特に、前回発表いたしました拙著「PC-8001マシン語活用ハンドブック」に対して多くの読者より寄せられました、システム・サブルーチンの使用例をより豊富に、という御意見に対しては充分に配慮し、解り易い構成を心がけさせていただいたつもりです。

読者諸兄におかれては、本書を熟読されることにより、N88-BASICの内部構造を一層深く理解され、PC-8801の更に高度な活用をされるための、お役に立たん事を深く希望するものであります。

本書の執筆、出版に当たって一方ならぬお世話になりました、石井晴正氏、橋本正樹氏、小牧自行氏、芝田憲一氏、水上英樹氏、今井勇吉氏、渡部孝子氏に感謝いたします。

1982年11月1日

FORESIGHT企画部代表 川村 清

# 目 次

## 第1章 システム・サブルーチン SYSTEM SUBROUTINES .....1

0018H: 周辺機器への1バイト出力を行なう .....	4
0020H: レジスタ・ペアの比較を行なう .....	6
0028H: FACCの符号を調べる .....	8
0030H: FACCの型を調べる .....	10
03B3H: エラーを出力してOSに制御を移す .....	12
05BDH: リンク・ポインタをセットする .....	14
0605H: テキスト中から特定のラインをさがし出す .....	15
1DE6H: 単精度型実数の減算を行なう .....	18
1DEAH: 単精度型実数の加算を行なう .....	20
1F10H: 単精度型実数の自然対数を求める .....	22
1F53H: 単精度型実数の乗算を行なう .....	24
1FB7H: 単精度型実数の除算を行なう .....	26
20ABH: FACCの符号を反転する .....	28
20E8H: 単精度型実数の移動を行なう .....	30
2134H: 単精度型実数の比較を行なう .....	32
215FH: 整数の比較を行なう .....	34
21A0H: FACCのデータを整数に変換する .....	36
21BCH: 単精度型実数を整数に変換する .....	38
21FDH: 整数をFACCに格納する .....	40
2202H: FACCの型を指定する .....	42
2214H: FACCのデータを単精度型に変換する .....	46
221CH: 倍精度型実数を単精度型実数に変換する .....	48
222FH: 整数を単精度型実数に変換する .....	50
223EH: FACCのデータを倍精度型実数に変換する .....	52
2246H: 単精度型実数を倍精度型実数に変換する .....	54
2252H: FACCの型を単精度実数型に指定する .....	56

2 3 2 F H : 整数の減算を行なう	58
2 3 3 A H : 整数の加算を行なう	60
2 3 5 A H : 整数の乗算を行なう	62
2 3 A B H : 整数の除算を行なう	64
2 3 F 7 H : 整数の符号を反転する	66
2 3 F A H : 整数の符号を反転する	68
2 4 0 C H : 整数の除算を行ない剰余を求める	70
2 4 1 D H : 倍精度型実数の減算を行なう	72
2 4 2 4 H : 倍精度型実数の加算を行なう	74
2 5 5 3 H : 倍精度型実数の乗算を行なう	76
2 6 2 9 H : 倍精度型実数の除算を行なう	78
2 6 B 5 H : 文字列を倍精度型実数に変換する	80
2 6 B C H : 文字列を数値データに変換する	82
2 8 C 2 H : 符号無し 16 ビットの 10 進数を出力する	84
2 8 D 0 H : 数値データを文字列に変換する	86
2 8 D 1 H : フォーマットを付けて文字列に変換する	88
2 E 0 5 H : 単精度型実数の平方根を求める	94
2 E 1 5 H : 単精度の指数演算を行なう	96
2 E 6 E H : e に対する指数関数の値を求める	98
2 F 1 A H : 単精度実数型の乱数を発生する	100
2 F 8 B H : 単精度実数型の余弦 (コサイン) を求める	102
2 F 9 1 H : 単精度実数型の正弦 (サイン) を求める	104
3 0 2 C H : 正接値 (タンジェント) を求める	106
3 5 8 3 H : キーボード 1 文字入力待ちを行なう	108
3 5 C 2 H : ストップ・キーのチェックを行なう	109
3 5 C E H : リアルタイムなキー入力を行なう	111
3 6 9 A H : ディスクと 1 セクタの入出力を行なう	113
3 6 E 2 H : P C - 8 0 3 1 のイニシアライズを行なう	116

37C9H:PC-8031へコマンドを送信する .....	117
37D2H:PC-8031へデータを送信する .....	118
3847H:PC-8031からのデータ受信を行なう .....	120
3DCBH:ディスク・ドライブのタイプを調べる .....	122
3DD9H:PC-8031のドライブ・ナンバを求める .....	124
3E0DH:CRTスクリーン1バイト出力を行なう .....	126
3E0H:内蔵ブザーをコントロールする .....	127
3ED4H:プリンタへの1バイト出力を行なう .....	128
4047H:タイマからデータの読み込みを行なう .....	129
428BH:カーソルを消去する .....	132
4290H:カーソルの表示を行なう .....	134
429DH:座標からVRAMのアドレスを求める .....	136
4350H:VRAM転送および属性設定を行なう .....	138
4452H:VRAM上のキャラクタを調べる.....	140
4472H:スクリーン上のキャラクタを調べる .....	142
447DH:カーソル移動および一文字表示を行なう .....	144
4B1AH:ホット・スタートを行なう .....	146
5FC8H:ライン・インプットを行なう .....	148
5F92H:スクリーン・エディットを行なう .....	150
6F6BH:CRTスクリーンのモード設定を行なう .....	154
71D9H:タイマヘデータの書き込みを行なう .....	157
72CDH:I/OをN <sub>88</sub> -BASIC用に設定する .....	160
780FH:ホット・スタートを行なう(2) .....	161
7ED0H:CMTを入力用にイニシアライズする .....	163
7F15H:CMTからの入力をクローズする .....	163
7F1AH:CMTへの出力をクローズする .....	163
7F35H:内蔵リレーのコントロールを行なう .....	164
7F4DH:CMTを出力用にイニシアライズする .....	166

7 F 8 7 H : CMTから1バイト入力を行なう .....	166
7 F D 0 H : CMTへの1バイト出力を行なう .....	166

## 第2章 I/Oポート・マップ I/OPORT MAPPING .....167

キー・マトリクスのデータを入力する .....	170
プリンタおよびタイマヘデータを出力する .....	173
8 2 5 1からデータを入力する .....	175
8 2 5 1ヘデータを出力する .....	175
8 2 5 1からステータスを入力する .....	176
8 2 5 1ヘコントロール・ワードを出力する .....	177
ディップ・スイッチのステータスを入力する .....	179
各種のコントロール信号を出力する .....	180
ディップ・スイッチのステータスを入力する .....	181
CRTおよびメモリのモード設定を行なう .....	182
各種のコントロール信号を入力する .....	183
各種のコントロール信号を出力する .....	184
CRTコントローラヘパラメータを出力する .....	185
CRTコントローラヘコマンドを出力する .....	185
ボーダー・カラー等を設定する .....	186
表示するGVRAMを指定する .....	187
カラー・パレットを設定する .....	188
GVRAMのステータスを入力する .....	190
GVRAMをセレクトする .....	191
DMAコントローラを制御する .....	193
オフセット・アドレスを入力する .....	194
オフセット・アドレス・レジスタを設定する .....	195
拡張ROMのステータスを入力する .....	196
拡張ROMをセレクトする .....	197

オフセット・アドレスをインクリメントする .....	198
8214のカレント・レジスタを設定する .....	201
インタラプトをコントロールする .....	201
漢字フォントを入力する .....	202
漢字ROMのアドレスを指定する .....	204
漢字ROMの読み出しを宣言する .....	206
PC-8031からデータを入力する .....	210
PC-8031へデータを出力する .....	211
PC-8031から制御信号を入力する .....	212
PC-8031へ制御信号を出力する .....	213

### 第3章 システム・ワーク・エリア SYSTEM WORKING AREA ..... 215

### 第4章 全回路図集 PC-8801 CIRCUIT DIAGRAM..... 233

1. CPU(1) .....	236
2. CPU(2) .....	237
3. ROM(1) .....	238
4. ROM(2) .....	239
5. RAM(1) .....	240
6. RAM(2) .....	241
7. I/O PORT(1) .....	242
8. I/O PORT(2) .....	243
9. INTERRUPT CONTROL(1) .....	244
10. INTERRUPT CONTROL(2) .....	245
11. FDC PORT(1) .....	246
12. FDC PORT(2) .....	247
13. SERIAL INTERFACE(1) .....	248
14. SERIAL INTERFACE(2) .....	249



15. KEYBOARD(1) .....	250
16. KEYBOARD(2) .....	251
17. CRT C(1) .....	252
18. CRT C(2) .....	253
19. GVRAM ADDRESS CONTROL(1) .....	254
20. GVRAM ADDRESS CONTROL(2) .....	255
21. GVRAM0 AND GVRAM1(1) .....	256
22. GVRAM0 AND GVRAM1(2) .....	257
23. GVRAM2(1) .....	258
24. GVRAM2(2) .....	259
25. COLOR CONTROL(1) .....	260
26. COLOR CONTROL(2) .....	261
27. VIDEO(1) .....	262
28. VIDEO(2) .....	263
29. BUS SLOT(1) .....	264
30. BUS SLOT(2) .....	265
31. DMAC .....	266

## 第5章 付章 APPENDIX.....267

A. 中間言語、処理アドレス一覧表 .....	269
B. 機械語モニタ処理アドレス一覧表 .....	274
C. $\mu$ PD3301アトリビュート・コード表 .....	275
D. キャラクタ・コード表 .....	276

## $\mu$ COM-82 インストラクション活用表.....277

1. $\mu$ COM-82 インストラクション・セット .....	279
2. $\mu$ COM-82 機械語 $\leftrightarrow$ ニーモニック対応表 .....	288
3. $\mu$ COM-82 ニーモニック $\leftrightarrow$ 機械語対照表 .....	291

## SAMPLE PROGRAMS

DISPLAY OR PRINT OUT .....	4
DUMP MEMORY .....	7
CHECK SIGN OF FLOATING NUMBER .....	9
CHECK TYPE OF FACC .....	10
OUTPUT ERROR OF N88-BASIC .....	12
SEARCH TEXT AND SET LINK POINTER .....	14
SUBTRACTION FOR SINGLE PRICISION .....	18
ADDITION FOR SINGLE PRECISION .....	20
LOG FUNCTION FOR SINGLE PRCISION .....	22
MULTIPLICATION FOR SINGLE PRECISION .....	24
DIVISION FOR SINGLE PRECISION .....	26
NEGATION FOR THE REAL NUMBER .....	28
DUMP SINGLE PRECISION NUMBER ON MEMORY .....	30
COMPARE FOR SINGLE PRECISION .....	32
COMPARE FOR INTEGER .....	34
CONVERT FACC INTO INTEGER .....	37
CONVERT SINGLE PRICISION NUMBER INTO INTEGER .....	38
NUMBERS WITH A BYTE .....	40
CONVERT ON MEMORY CODE INTO THE NUMBER .....	42
CONVERT FACC INTO SINGLE PRECISION .....	47
CONVERT DOUBLE PRECISION INTO SINGLE PRECISION .....	48
CONVERT INTEGER INTO SINGLE PRECISION NUMBER .....	50
CONVERT FACC INTO DOUBLE PRECISION .....	52
CONVERT SINGLE PRECISION INTO DOUBLE PRECISION .....	54
CALCULATE $R^2 * PI$ .....	56
SUBTRACTION FOR INTEGER .....	58
ADDITION FOR INTEGER .....	60
MULTIPLICATION FOR INTEGER .....	62
DIVISION FOR INTEGER .....	64
NEGATION FOR INTEGER .....	66
NEGATION FOR INTEGER .....	68
CALCULATE REMAINDER .....	70
SUBTRACTION FOR DOUBLE PRECISION .....	72
ADDITION FOR DOUBLE PRECISION .....	74
MULTIPLICATION FOR DOUBLE PRECISION .....	76
DIVISION FOR DOUBLE PRECISION .....	78
INPUT AND DISPLAY NUMBER .....	80
INPUT AND DISPLAY NUMBER .....	82
DISPLAY NUMBERS WITH TWO BYTES .....	84
INPUT AND DISPLAY A NUMBER FREE FORMAT .....	87
PRINT USING TEST PROGRAM .....	89
SQR FUNCTION TEST PROGRAM .....	94
EXPONENTIATION FOR SINGLE PRECISION .....	96



EXP FUNCTION TEST PROGRAM.....	98
DISPLAY RANDOM NUMBERS .....	101
COS FUNCTION TEST PROGRAM.....	102
SIN FUNCTION TEST PROGRAM .....	104
TAN FUNCTION TEST PROGRAM.....	106
INPUT AND DISPLAY CHARACTER CODE .....	108
CHECK STOP KEY TEST PROGRAM.....	110
REAL TIME KEY SCANNING .....	112
CONTROL DISK.....	113
COUNT CONNECTED DRIVES OF PC-8031 .....	116
INITIALIZE PC-8031.....	117
SET UP ONLY DRIVE 2 FOR SINGLE SURFACE .....	119
CHECK THE UNIT PC-8031 OR PC-8031-2W.....	121
CHECK TYPE OF DRIVE.....	122
CALCULATE DRIVE NUMBER OF PC-8031 .....	125
DISPLAY CHARACTERS .....	126
AMERICAN PATROL SIREN .....	127
OUTPUT MESSAGE TO PRINTER .....	128
READ FROM TIMER .....	129
CURSOR ON AND OFF.....	132
INPUT ROUTINE FOR BUSINESS PROGRAM .....	134
DISPLAY A BOX WITH CHARACTER .....	137
USE SEMI-GRAPHIC .....	139
ERASE NOT WHITE CHARACTERS .....	140
CHANGE SPACES INTO POINTS ON SCREEN.....	142
DISPLAY STEPS ON SCREEN .....	145
CLEAR TEXT PROGRAM OR NOT.....	146
CONVERT STRING INTO CAPITALS .....	148
MICRO WORD PROCESSER.....	150
CHANGE CRT SCREEN MODE .....	155
WRITE TO TIMER .....	158
INITIALIZE I/O FOR N88-BASIC .....	160
HOT START OR RETURN .....	161
CONTROL MOTOR RELAY.....	164
CHECK KEY SCANNING PORT .....	170
PRINT OUT CHARACTERS.....	174
EXECUTE ON N88-BASIC MODE .....	186
EXECUTE ON PC-8801 N-BASIC MODE .....	186
COLOR PALETTE DEMONSTRATION FAST VERSION .....	189
TEST GRAPHIC VIDEO RAM.....	192
DUMP TEXT RAM AREA OF N88-BASIC .....	198
ACCESS KANJI ROM AND DISPLAY FONT .....	207
RECEIVE A DATA FROM PC-8031 SUBROUTINE.....	210
SEND A DATA OR COMMAND TO PC-8031 SUBROUTINE.....	211



## ***SYSTEM SUBROUTINES***



本章は、N88-BASICのメインROMに含まれている多くの有用なシステム・サブルーチンの中から、一般ユーザの利用頻度が高いと思われるものを選んで、資料集としてまとめたものです。

本章の各ルーチンは、エントリ・アドレスによってソーティングし、分類しました。

<b>アドレス</b>	各ルーチンのエントリ・ポイントを示します。
<b>機能</b>	各ルーチンの実行目的を簡単に示します。
<b>レジスタ</b>	各ルーチンの実行後、内容が変更されるレジスタを示します。したがって、これらのレジスタの内容を保存しなければならない場合には、ユーザ・プログラム中でのレジスタ退避が必要です。なお、N88-BASICインタプリタでは、アルタネーティブ・レジスタおよびインデックス・レジスタを使用していません。
<b>解説</b>	各ルーチンの詳しい使用方法、入出力パラメータなどを説明した上で、場合によっては関連して予備知識も示します。
<b>サンプル</b>	各ルーチンを利用したサンプル・プログラムを全てのルーチンに対して示しておきました。これらのサンプル・プログラムは一部を除いて、機械語のサブルーチン形式となっており、実行例も加えてあります。 なお、ニーモニックはザイログ形式のZ80 ( $\mu$ PD780) ニーモニックであるため、N88-BASICモニタの内蔵アセンブラによって直接アセンブルする事はできません。もちろんオブジェクト・コードを、モニタのSコマンドまたはEコマンドによって入力すればサンプル・プログラムを実行する事ができます。

本章の執筆にあたっては次の構成のシステムを使用しました。

PC-8801	シリアル・ナンバ2803521BS
PC-8801-01	PC-8801用漢字ROMボード
PC-8822	18ピン・トッド・マトリクス漢字プリンタ
PC-8031-2W	両面倍密度デュアル・ミニ・ディスク・ユニット
PC-8034-2W	PC-8031-2W用システム・ディスク
PC-8046	9インチ・グリーン・ディスプレイ。

## 0018H:周辺機器への1バイト出力を行なう。

アドレス	0018H
機能	周辺機器への1バイト出力を行なう。
レジスタ	全て保存

**解説** E64CH番地に格納されているデータが00Hの場合にはCRTスクリーンへ、01Hの場合にはプリンタへ、それぞれアキュムレータのデータ1バイトを出力します。

通常は、E64CH番地に00Hを与えて、CRTスクリーンへの1文字出力ルーチンとして利用できます。

### サンプル

```

;
; --- display or print out ---
;
; ORG 0B900H
B900 AF START: XOR A
B901 324CE6 LD (0E64CH),A ; select crt for output
;
B904 215EB9 LD HL,PROMPT
B907 CD55B9 CALL DSPMSG
B90A CD8335 CALL 3583H ; input a character from keyboard
B90D FE03 CP 03H
B90F C8 RET Z ; stop key, return to basic
;
B910 F5 PUSH AF
B911 CD0D3E CALL 3E0DH
B914 3E0D LD A,0DH
B916 CD0D3E CALL 3E0DH ; echo-back and line feed on crt
B919 3E0A LD A,0AH
B91B CD0D3E CALL 3E0DH
B91E F1 POP AF
;
B91F FE63 CP 'c'
B921 2829 JR Z,DISPLAY
B923 FE43 CP 'C'
B925 2825 JR Z,DISPLAY
B927 FE64 CP 'd'
B929 2821 JR Z,DISPLAY
B92B FE44 CP 'D'
B92D 281D JR Z,DISPLAY
;
B92F FE6C CP 'l'
B931 280E JR Z,LPRINT
B933 FE4C CP 'L'
B935 280A JR Z,LPRINT
B937 FE70 CP 'p'
B939 2806 JR Z,LPRINT
B93B FE50 CP 'P'
B93D 2802 JR Z,LPRINT
;
B93F 18BF JR START
;
B941 2175B9 LPRINT:LD HL,CARE
B944 CD55B9 CALL DSPMSG
;
B947 3E01 LD A,1

```

```

B949 324CE6      LD    (0E64CH),A      ; select printer for output
;
; main section
;
B94C 3E20      DISPLAY:LD    A,20H
;
B94E DF        LOOP: RST    18H      ;*output a character
B94F 3C        INC    A
B950 FE00      CP    0FFH+1
B952 20FA      JR    NZ,LOOP
;
B954 C9        RET      ; return to basic
;
; display message subroutine
;
B955 7E        DSPMSG:LD    A,(HL)
B956 A7        AND    A
B957 C8        RET    Z
B958 CD0D3E    CALL 3E0DH      ; display a character
B95B 23        INC    HL
B95C 18F7      JR    DSPMSG
;
; message area
;
B95E 63727420  PROMPT:DEFB 'crt or line-printer ? ',0
B962 6F72206C
B966 696E652D
B96A 7072696E
B96E 74657220
B972 3F2000
B975 6C696E65  CARE:  DEFB 'line-printer active !!',0AH,0DH,0
B979 2D707269
B97D 6E746572
B981 20616374
B985 6976652D
B989 21210A0D
B98D 00
;

```

```

a=&hb900:call a
crt or line-printer ? c
! '*$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN0PQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
pqrstuvwxy{!}"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN0PQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
タチツトナニヌネノハヒフヘホマミメモヤヨヲリルレロクン*==≡≡▲▼◆♦◆●○×/◇◇◇◇◇◇◇◇
Ok
call a
crt or line-printer ? l
line-printer active !!
Ok

```

## 0020H：レジスタ・ペアの比較を行なう。

アドレス	0020H
機能	レジスタ・ペアの比較を行なう。
レジスタ	A, F

**解 説** レジスタHLに格納された16ビット符号無し of 整数と、レジスタDEに格納された16ビット符号無し of 整数を比較し、結果をゼロ・フラグ(Z)およびキャリ・フラグ(CY)に入れてもどります。

それぞれのフラグは、レジスタHLとレジスタDEのデータが等しい場合にはゼロ・フラグ(Z)を1にセットし、前者の方が小さい場合にはキャリ・フラグ(CY)を1にセットします。



# サンプル

```

;
; --- dump memory ---
;
; ORG 0B900H
B900 110000 LD DE,0000H ; start address for dump memory
B903 217F00 LD HL,007FH ; end address for dump memory
;
; DUMP: LD A,D
B907 CD23B9 CALL DSPACC
B90A 7B LD A,E
B90B CD23B9 CALL DSPACC
;
B90E 3E3A LD A,' '
B910 CD0D3E CALL 3E0DH
;
B913 1A LD A,(DE)
B914 CD23B9 CALL DSPACC
;
B917 3E20 LD A,' '
B919 CD0D3E CALL 3E0DH
;
B91C 13 INC DE
;
B91D CD2000 CALL 0020H ;*compare register hl-de
B920 30E4 JR NC,DUMP
;
B922 C9 RET ; return to basic
;
; display accumulator hexa decimal subroutine
;
; destroys : register a,f,b
;
B923 47 DSPACC:LD B,A
B924 0F RRCA
B925 0F RRCA
B926 0F RRCA
B927 0F RRCA
B928 CD2CB9 CALL HALF
B92B 78 LD A,B
;
B92C E60F HALF: AND 0FH
B92E FE0A CP 10
B930 3802 JR C,NUMBER
B932 C607 ADD A,7
;
B934 C630 NUMBER:ADD A,30H
B936 CD0D3E CALL 3E0DH ; display a character
B939 C9 RET
;

```

```

a=&hb900:call a
0000:F3 0001:31 0002:A0 0003:E1 0004:C3 0005:F7 0006:77 0007:00 0008:7E 0009:E3
000A:BE 000B:23 000C:E3 000D:C2 000E:93 000F:03 0010:23 0011:7E 0012:FE 0013:3A
0014:D0 0015:C3 0016:15 0017:0A 0018:F5 0019:CD 001A:42 001B:ED 001C:C3 001D:25
001E:59 001F:00 0020:7C 0021:92 0022:C0 0023:7D 0024:93 0025:C9 0026:00 0027:00
0028:3A 0029:44 002A:EC 002B:B7 002C:C2 002D:83 002E:20 002F:C9 0030:3A 0031:BD
0032:EA 0033:FE 0034:08 0035:C3 0036:27 0037:15 0038:C3 0039:69 003A:E6 003B:00
003C:00 003D:00 003E:00 003F:00 0040:A0 0041:21 0042:FD 0043:21 0044:7E 0045:A7
0046:C8 0047:FE 0048:20 0049:C9 004A:21 004B:FF 004C:FF 004D:22 004E:56 004F:E6
0050:23 0051:C9 0052:22 0053:41 0054:EC 0055:C1 0056:C1 0057:4F 0058:78 0059:D3
005A:71 005B:79 005C:C1 005D:C3 005E:E9 005F:11 0060:CD 0061:BD 0062:05 0063:CD
0064:21 0065:4F 0066:C3 0067:D6 0068:4E 0069:CD 006A:9F 006B:ED 006C:F3 006D:3A
006E:C2 006F:E6 0070:C9 0071:00 0072:3A 0073:B9 0074:E6 0075:A7 0076:37 0077:C4
0078:21 0079:40 007A:C3 007B:6A 007C:6F 007D:3A 007E:E9 007F:E9
Ok

```

## 0028H: FACCの符号を調べる.

アドレス	0028H
機能	フローティング・アキュムレータ (FACC) の符号を調べる.
レジスタ	A, F

**解 説** フローティング・アキュムレータ (FACC) に格納されている単精度型実数または倍精度型実数の符号を調べ、結果をアキュムレータに与えてもどります。

フローティング・アキュムレータ (FACC) の符号とアキュムレータに与える数値の関係は次のとおりです。

FACCの実数が負の場合アキュムレータにFFHを与えます。

FACCの実数が0の場合アキュムレータに00Hを与えます。

FACCの実数が正の場合アキュムレータに01Hを与えます。

ただし、フローティング・アキュムレータ (FACC) に格納されている数値は必ず実数型でなければならず、整数型では正常な結果が得られません。これは、フローティング・アキュムレータ (FACC) の最上位バイト (EC44H番地) の内容によって符号を調べているからです。

[illegible]

## 0030H: FACCの型を調べる.

アドレス	0030H
機能	フローティング・アキュムレータ (FACC) の型を調べる.
レジスタ	A, F

**解説** EABDH番地に格納されているフローティング・ポイント・アキュムレータの型を表わす数値をチェックして、サイン・フラグ (S)、ゼロ・フラグ (Z)、パリティ/オーバーフロー・フラグ (P/V)、およびキャリ・フラグ (CY) を次のようにセットしてもどります。

- S ← 整数型 (2) でセットします。  
 Z ← 文字型 (3) でセットします。  
 P/V ← 単精度実数型 (4) でパリティ・オッドを指定します。  
 CY ← 倍精度実数型 (8) でリセットします。

### サンプル

```

;
; --- check type of facc ---
;
;      ORG 0B900H
B900 CD3000      ;      CALL 0030H          ;check type of facc
;
B903 F20BB9      ;      JP P,NOT2
B906 212BB9      LD HL,TYPE2          ; integer
B909 1817        JR DSPMSG
;
B90B 2005      NOT2: JR NZ,NOT3
B90D 2136B9     LD HL,TYPE3          ; string
B910 1810      JR DSPMSG
;
B912 EA1AB9     NOT3: JP PE,NOT4
B915 2140B9     LD HL,TYPE4          ; single precision number
B918 1808      JR DSPMSG
;
B91A 3805     NOT4: JR C,ERROR
B91C 215BB9     LD HL,TYPE8          ; double precision number
B91F 1801      JR DSPMSG
;
B921 C9        ERROR: RET          ; error, return to basic
;
; display message section
;
B922 7E        DSPMSG:LD A,(HL)
B923 A7        AND A
B924 C8        RET Z                ; return to basic
B925 CD0D3E     CALL 3E0DH          ; display a character
B928 23        INC HL
B929 18F7      JR DSPMSG
;
; message area
;
B92B 696E7465   TYPE2: DEFB 'integer !!',0
B92F 67657220
B933 212100
B936 73747269   TYPE3: DEFB 'string !!',0

```

```

B93A 6E672021
B93E 2100
B940 73696E67 TYPE4: DEFB 'single precision number !!',0
B944 6C652070
B948 72656369
B94C 73696F6E
B950 206E756D
B954 62657220
B958 212100
B95B 646F7562 TYPE8: DEFB 'double precision number !!',0
B95F 6C652070
B963 72656369
B967 73696F6E
B96B 206E756D
B96F 62657220
B973 212100
;

```

```

def usr=&hb900:a%=usr(a%)
integer !!
Ok
def usr=&hb900:a$=usr(a$)
string !!
Ok
def usr=&hb900:a!=usr(a!)
single precision number !!
Ok
def usr=&hb900:a#=usr(a#)
double precision number !!
Ok

```

## 03B3H: エラーを出力してOSに制御を移す.

**アドレス** 03B3H

**機能** エラーを出力してOSに制御を移す.

**レジスタ**

**解説** レジスタEのエラー・コードに対応するエラーを出力して、N88-BASICの制御下に入ります.

ただし、レジスタEのエラー・コードに対応するエラーが存在しない場合には、“Unprintable error” (UE Error) エラーを出力します.

また、エラー・コードが00H (ゼロ) の場合には、無限ループに入り込んでしまうため注意が必要です.

### サンプル

```

;
; --- output error of n88-basic ---
;
;          ORG 0B900H
B900 2137B9      LD HL,PROMPT          ; error code ?
B903 CD2EB9      CALL DSPMSG
;
B906 1E00        LD E,0              ; initialize input error code
B908 3E30        LD A,'0'           ; initialize input character
;
B90A F5          INLOOP:PUSH AF      ;
B90B 7B          LD A,E              ;
B90C 87          ADD A,A             ;
B90D 87          ADD A,A             ;
B90E 83          ADD A,E             ;
B90F 87          ADD A,A             ; e <-- e * 10 + input number
B910 5F          LD E,A              ;
B911 F1          POP AF              ;
B912 D630        SUB '0'             ;
B914 83          ADD A,E             ;
B915 5F          LD E,A              ;
;
B916 CD8335      CALL 3583H           ; input a character from keyboard
B919 CD0D3E      CALL 3E0DH           ; echo-back a character
;
B91C FE03        CP 03H              ;
B91E C8          RET Z               ; stop key, return to basic
;
B91F FE0D        CP 0DH              ;
B921 20E7        JR NZ,INLOOP        ;
;
B923 3E0A        LD A,0AH            ; line feed code in accumulator
B925 CD0D3E      CALL 3E0DH           ; display a character
;
B928 1C          INC E               ;
B929 1D          DEC E               ;
B92A C8          RET Z               ;
;
B92B C3B303      JP 03B3H             ;*output error and jump to basic
;
; display message subroutine
;
B92E 7E          DSPMSG:LD A,(HL)

```

```

B92F A7          AND A
B930 C8          RET Z
B931 CD003E      CALL 3E00H      ; display a character
B934 23          INC HL
B935 18F7        JR  DSPMSG
                ;
                ; message area
                ;
B937 6572726F    PROMPT:DEFB 'error code ? ',0
B93B 7220636F
B93F 6465203F
B943 2000
                ;

```

```

a=&hb900:call a
error code ? 0
Ok
call a
error code ? 1
NEXT without FOR
Ok
call a
error code ? 2
Syntax error
Ok
call a
error code ? 3
RETURN without GOSUB
Ok
call a
error code ? 10
Duplicate Definition
Ok
call a
error code ? 11
Division by zero
Ok
call a
error code ? 40
Unprintable error
Ok
call a
error code ? 41
Unprintable error
Ok

```



## 05BDH:リンク・ポインタをセットする。

アドレス	05BDH
機能	リンク・ポインタをセットする。
レジスタ	A, F, B, C, D, E, H, L

**解 説** E658H～E659H番地に格納されているプログラム・エリアの先頭アドレス（通常は0001H）からレジスタHLをポインタとして、プログラム・エリアを上位に向かってサーチして行き、次の行の先頭アドレスを見つけたら現在行のリンク・ポインタに代入する作業を、次行の先頭2バイトが00Hになるまで続けます。

終了時には、レジスタHLにプログラムのエンド・アドレスが、レジスタDEにはレジスタHLの値から1を減じたものが入っています。

### サンプル

```

;
; --- search text and set link pointer ---
;
      ORG 0B900H
;
B900 C0B005      CALL 05BDH          ;*search text and set link pointer
B903 210FB9      LD HL,MESAGE
B906 7E          DSPMSG:LD A,(HL)
B907 A7          AND A
B908 C8          RET Z
B909 CD0D3E      CALL 3E0DH          ; display a character
B90C 23          INC HL
B90D 18F7        JR DSPMSG
;
; message area
;
B90F 6E38382D    MESSAGE:DEFB 'n88-basic text linked !!',0
B913 62617369
B917 63207465
B91B 7874206C
B91F 696E6B65
B923 64202121
B927 00
;

```

```

a=&hb900:call a
n88-basic text linked !!
Ok

```



## 0605H：テキスト中から特定のラインをさがし出す。

アドレス	0605H
機能	テキスト中から特定のラインをさがし出す。
レジスタ	A, F, B, C, H, L

**解説** N88-BASICのテキスト・エリア中から、レジスタDEで指定するライン・ナンバのラインをさがし出すためのサブルーチンです。実行結果はレジスタBCおよびレジスタHLにウィンド中のアドレス(8000H~83FH)として与えられ、そのライン・ナンバを含むブロックがウィンドに表われますが、ゼロ・フラグ(Z)とキャリ・フラグ(CY)の変化によって次に示す3通りの実行結果に分かれます。

- 1) ゼロ・フラグ(Z)、キャリ・フラグ(CY)共に、セットされた場合、指定したライン・ナンバが存在する事を示します。レジスタBCには指定ラインの先頭(リンク・ポインタも含む)アドレスが、レジスタHLには指定した次のラインの先頭アドレスが与えられます。
- 2) ゼロ・フラグ(Z)がセットされ、キャリ・フラグ(CY)がリセットされた場合には、指定したライン・ナンバがテキスト中に存在せず、なおかつ、指定ライン・ナンバより大きいナンバのラインが1行も存在しない事を示します。レジスタBCおよびレジスタHLには、N88-BASICのテキスト・エンドを表わす2バイトの00Hが格納されている最初のアドレスが与えられます。
- 3) ゼロ・フラグ(Z)、キャリ・フラグ(CY)共にリセットされた場合には、指定したライン・ナンバがテキスト中に存在しない事を示しますが、レジスタBCには指定したナンバより大きい最初のラインの先頭アドレスが、レジスタHLにはその次のラインの先頭アドレスが与えられます。

### サンプル

```

; --- search the basic text for the line ---
;
;      ORG 0B900H
;
B900 2162B9      LD HL,PROMPT          ; line number ?
B903 CD59B9      CALL DSPMSG
;
B906 210000      LD HL,0
B909 3E30        LD A,'0'
;
B90B E5          INLOOP:PUSH HL
```

```

B90C D1      POP    DE
B90D 29      ADD    HL,HL
B90E 29      ADD    HL,HL
B90F 19      ADD    HL,DE
B910 29      ADD    HL,HL
B911 D630    SUB    '0'
B913 5F      LD     E,A
B914 1600    LD     D,0
B916 19      ADD    HL,DE

;
B917 CD8335  CALL    3583H      ; input a character from keyboard
B91A CD0D3E  CALL    3E0DH      ; echo-back a character
B91D FE0D    CP      0DH
B91F 20EA    JR      NZ,INLOOP

;
B921 EB      EX      DE,HL      ; set line number in register de
B922 CD0506  CALL    0605H      ; *search the text for the line
B925 300F    JR      NC,ERROR   ; reset cy, not found

;
B927 218DB9  LD      HL,FOUND   ; line from
B92A CD59B9  CALL    DSPMSG
B92D CD3DB9  CALL    DSPBC     ; display address
B930 3E48    LD      A,'H'
B932 CD0D3E  CALL    3E0DH      ; display a character
B935 C9      RET              ; return to basic

;
B936 2171B9  ERROR: LD    HL,NFOUND ; line number not found !!
B939 CD59B9  CALL    DSPMSG
B93C C9      RET              ; return to basic

;
; display register bc hexa decimal subroutine
;
; destroys : register a,f,b
;
B93D 78      DSPBC: LD    A,B
B93E CD42B9  CALL    DSPACC
B941 79      LD      A,C

;
B942 47      DSPACC:LD    B,A
B943 0F      RRCA
B944 0F      RRCA
B945 0F      RRCA
B946 0F      RRCA
B947 CD48B9  CALL    HALF
B94A 78      LD      A,B

;
B94B E60F    HALF: AND    0FH
B94D FE0A    CP      10
B94F 3802    JR      C,NUMBER
B951 C607    ADD     A,7

;
B953 C630    NUMBER:ADD   A,30H
B955 CD0D3E  CALL    3E0DH      ; display a character
B958 C9      RET

;
; display message subroutine
;
; destroys : register a,f,h,l
;
B959 7E      DSPMSG:LD    A,(HL)
B95A A7      AND     A
B95B C8      RET     Z
B95C CD0D3E  CALL    3E0DH      ; display a character
B95F 23      INC     HL
B960 18F7    JR      DSPMSG

;
; message area
;
B962 6C696E65 PROMPT:DEFB 'line number ? ',0
B966 206E756D
B96A 62657220
B96E 3F2000
B971 0D0A6C69 NFOUND:DEFB 0DH,0AH,'line number not found !!',07H,0
B975 6E65206E

```

```

B979 756D6265
B97D 72206E6F
B981 7420666F
B985 756E6420
B989 21210700
B98D 0D0A6C69 FOUND: DEFB 0DH,0AH,'line from ',0
B991 6E652066
B995 726F6D20
B999 00

```

;

```

list
100 PRINT 'line number 100'
110 PRINT 'line number 110'
120 PRINT 'line number 120'
130 PRINT 'line number 130'
140 PRINT 'line number 140'
150 PRINT 'line number 150'
Ok
a=&hb900:call a
line number ? 10
line number not found !!
Ok
call a
line number ? 100
line from 8001H
Ok
call a
line number ? 110
line from 8019H
Ok
call a
line number ? 105
line number not found !!
Ok
call a
line number ? 150
line from 8079H
Ok
call a
line number ? 60000
line number not found !!
Ok
mon

```

```

hJd8000,80ff
8000 00 19 00 64 00 91 20 22 6C 69 6E 65 20 6E 75 6D
8010 62 65 72 20 31 30 30 22 00 31 00 6E 00 91 20 22
8020 6C 69 6E 65 20 6E 75 6D 62 65 72 20 31 31 30 22
8030 00 49 00 78 00 91 20 22 6C 69 6E 65 20 6E 75 6D
8040 62 65 72 20 31 32 30 22 00 61 00 82 00 91 20 22
8050 6C 69 6E 65 20 6E 75 6D 62 65 72 20 31 33 30 22
8060 00 79 00 8C 00 91 20 22 6C 69 6E 65 20 6E 75 6D
8070 62 65 72 20 31 34 30 22 00 91 00 96 00 91 20 22
8080 6C 69 6E 65 20 6E 75 6D 62 65 72 20 31 35 30 22
8090 00 00 00 64 28 FF 97 28 58 29 29 3B 00 A8 00 B4
80A0 00 20 20 20 20 20 83 00 B3 00 BE 00 20 20 20 20
80B0 20 9B 00 BA 00 C8 00 20 83 00 00 00 B4 20 20 8B
80C0 20 0C 80 00 F2 50 20 DD 20 48 57 24 F1 4B 57 24
80D0 F3 FF 96 28 50 F4 0C 80 00 29 3A 89 20 0E 0E 00
80E0 00 F6 00 0B 00 20 20 20 4B 57 24 F1 4B 57 24 F3
80F0 FF 96 28 50 29 00 0F 01 0C 00 20 20 20 41 44 44
hJ^b
Ok

```

```

d t 'line num
ber 100' 1 n t
line number 110'
I x t 'line num
ber 120' a = t
line number 130'
y t 'line num
ber 140' t l t
line number 150'
d( KX)); i I
= u se
J コ ネ ■ I ■
一 年 ン KW$月KW$
月 I(PB - ):!!
カ KW$月KW$月
I(P) ADD

```

# 1DE6H: 単精度型実数の減算を行なう.

**アドレス** 1DE6H

**機能** 単精度型実数の減算を行なう.

**レジスタ** A, F, B, C, D, E, H, L

**解説** レジスタBCDEに格納された単精度型の実数を被減数とし、フローティング・アキュムレータに格納された単精度型の実数を減数として減算を行ない、フローティング・アキュムレータに差を格納してもどります.

## サンプル

```

;
; --- subtraction for single precision ---
;
        ORG 0B900H
;
START: LD  HL,PRMPT1
        CALL DSPMSG
        CALL 5F92H                ; screen editor
        RET C                    ; stop key, return to basic
        INC HL
        CALL CLRFBAC              ; clear floating point accumulator
        CALL 26C8H                ; convert into binary code
        CALL 2214H                ; convert into single precision
        CALL 20E8H                ; move from facc to register bcde
        PUSH BC
        PUSH DE
;
B919 216AB9 LD  HL,PRMPT2
B91C CD53B9 CALL DSPMSG
B91F CD925F CALL 5F92H                ; screen editor
B922 23      INC HL
B923 CD41B9 CALL CLRFBAC              ; clear floating point accumulator
B926 CDC826 CALL 26C8H                ; convert into binary code
B929 CD1422 CALL 2214H                ; convert into single precision
;
B92C D1      POP DE
B92D C1      POP BC
B92E CDE61D CALL 1DE6H                ; subtract [facc] from [reg-bcde]
B931 CDD028 CALL 28D0H                ; convert into character code
B934 E5      PUSH HL
B935 2175B9 LD  HL,ANSWER
B938 CD53B9 CALL DSPMSG
B93B E1      POP HL
B93C CD53B9 CALL DSPMSG                ; display result
;
B93F 18BF JR  START
;
; clear floating point accumulator subroutine
;
CLRFBAC: PUSH HL
        LD  HL,0
        LD  (0EC3DH),HL
        LD  (0EC3FH),HL
        LD  (0EC41H),HL
        LD  (0EC43H),HL
        POP HL
        RET
;
; display message subroutine
;
B953 7E DSPMSG: LD  A,(HL)

```

```

B954 A7          AND  A
B955 C8          RET  Z
B956 C00D3E      CALL 3E0DH          ; display a character
B959 23          INC  HL
B95A 18F7        JR   DSPMSG

```

```

;
; message area

```

```

B95C 0D0A0A20    PRMPT1:DEFB 0DH,0AH,0AH,'      x  ? ',0
B960 20202078
B964 2020203F
B968 2000
B96A 20202020    PRMPT2:DEFB '      y  ? ',0
B96E 79202020
B972 3F2000
B975 78202D20    ANSWER:DEFB 'x - y --> ',0
B979 79202D20
B97D 3E2000

```

```

;

```

```

a=&hb900:call a

```

```

      x  ? 20
      y  ? 30
x - y --> -10

```

```

      x  ? 1000000
      y  ? 1
x - y --> 999999

```

```

      x  ? 12345678901234567890
      y  ? 0
x - y --> 1.23457E+19

```

```

      x  ? 0.5
      y  ? 0.6
x - y --> -.1

```

```

      x  ?
Ok

```

# 1DEAH: 単精度型実数の加算を行なう.

**アドレス** 1DEAH

**機能** 単精度型実数の加算を行なう.

**レジスタ** A, F, B, C, D, E, H, L

**解説** レジスタBCDEに格納された単精度型実数に、フローティング・アキュムレータに格納された単精度型実数を加え、フローティング・アキュムレータに和を格納してもどります.

## サンプル

```

;
; --- addition for single precision ---
;
        ORG 0B900H
;
B900 215CB9  START: LD  HL,PRMPT1
B903 CD53B9      CALL DSPMSG
B906 CD925F      CALL 5F92H          ; screen editor
B909 D8          RET C              ; stop key, return to basic
B90A 23          INC HL
B90B CD41B9      CALL CLRFBAC       ; clear floating point accumulator
B90E CDC826      CALL 26C8H         ; convert into binary code
B911 CD1422      CALL 2214H         ; convert into single precision
B914 CDE820      CALL 20E8H         ; move from facc to register bcde
B917 C5          PUSH BC
B918 D5          PUSH DE
;
B919 216AB9      LD  HL,PRMPT2
B91C CD53B9      CALL DSPMSG
B91F CD925F      CALL 5F92H          ; screen editor
B922 23          INC HL
B923 CD41B9      CALL CLRFBAC       ; clear floating point accumulator
B926 CDC826      CALL 26C8H         ; convert into binary code
B929 CD1422      CALL 2214H         ; convert into single precision
;
B92C D1          POP  DE
B92D C1          POP  BC
B92E CDEA1D      CALL 1DEAH         ; add [register bcde] and [facc]
B931 CDD028      CALL 28D0H         ; convert into character code
B934 E5          PUSH HL
B935 2175B9      LD  HL,ANSWER
B938 CD53B9      CALL DSPMSG
B93B E1          POP  HL
B93C CD53B9      CALL DSPMSG       ; display result
;
B93F 18BF        JR   START
;
; clear floating point accumulator subroutine
;
B941 E5          CLRFBAC:PUSH HL
B942 210000      LD  HL,0
B945 223DEC      LD  (0EC3DH),HL
B948 223FEC      LD  (0EC3FH),HL
B94B 2241EC      LD  (0EC41H),HL
B94E 2243EC      LD  (0EC43H),HL
B951 E1          POP  HL
B952 C9          RET
;
; display message subroutine
;

```

```

B953 7E      DSPMSG:LD    A,(HL)
B954 A7      AND    A
B955 C8      RET    Z
B956 C0D03E  CALL  3E0DH      ; display a character
B959 23      INC    HL
B95A 18F7    JR     DSPMSG

```

```

;
; message area
;
B95C 00D0A02u PRMPT1:DEFB 0DH,0AH,0AH,' x ? ',0
B960 20202078
B964 2020203F
B968 2000
B96A 20202020 PRMPT2:DEFB ' y ? ',0
B96E 79202020
B972 3F2000
B975 78202B20 ANSWER:DEFB 'x + y --> ',0
B979 79202D20
B97D 3E2000
;

```

```

a=&hb900:call a

```

```

x ? 0.123
y ? 0.007
x + y --> .13

```

```

x ? 20
y ? -50
x + y --> -30

```

```

x ? 123.456
y ? 0.0000001
x + y --> 123.456

```

```

x ? abc
y ? def
x + y --> 0

```

```

x ?
Ok

```



## 1 F 1 0 H：単精度型実数の自然対数を求める。

アドレス	1 F 1 0 H
機 能	単精度型実数の自然対数を求める。
レジスタ	A, F, B, C, D, E, H, L

**解 説** フローティング・アキュムレータ (F A C C) に格納された単精度型実数の自然対数 (e を底とした対数) を、同じくフローティング・アキュムレータ (F A C C) に格納してもどります。

### サンプル

```

;
; --- log function for single precision ---
;
        ORG 0B900H
;
B900 214DB9  START: LD  HL,PROMPT
B903 CD44B9  CALL DSPMSG
;
B906 CD925F  CALL 5F92H          ; screen editor
B909 08      RET C          ; stop key, return to basic
B90A 23      INC HL
B90B E5      PUSH HL
B90C CD32B9  CALL CLRFBAC      ; clear floating point accumulator
B90F CDC826  CALL 26C8H      ; convert into binary code
B912 CD1422  CALL 2214H      ; convert into single precision
B915 CD101F  CALL 1F10H      ; *log function
B918 CDD028  CALL 28D0H      ; convert into character code
B91B EB      EX DE,HL        ; top address of result in reg-de
;
B91C 216AB9  LD HL,ANS1
B91F CD44B9  CALL DSPMSG
;
B922 E1      POP HL          ; top address of a input number
B923 CD44B9  CALL DSPMSG      ; display a input number again
;
B926 2171B9  LD HL,ANS2
B929 CD44B9  CALL DSPMSG
;
B92C EB      EX DE,HL        ; top address of result in reg-hl
B92D CD44B9  CALL DSPMSG      ; display result
;
B930 18CE    JR START
;
; clear floating point accumulator subroutine
B932 E5      CLRFBAC: PUSH HL
B933 210000  LD HL,0
B936 223DEC  LD (0EC3DH),HL
B939 223FEC  LD (0EC3FH),HL
B93C 2241EC  LD (0EC41H),HL
B93F 2243EC  LD (0EC43H),HL
B942 E1      POP HL
B943 C9      RET
;
; display message subroutine
;
B944 7E      DSPMSG: LD A,(HL)
B945 A7      AND A
B946 C8      RET Z
B947 CD0D3E  CALL 3E0DH      ; display a character
B94A 23      INC HL

```



```

B94B 18F7          JR    DSPMSG
          ;
          ; message area
          ;
B94D 0D0A7369  PROMPT:DEFB 0DH,0AH,'single precision number ? ',0
B951 6E676C65
B955 20707265
B959 63697369
B95D 6F6E206E
B961 756D6265
B965 72203F20
B969 00
B96A 6C6F6720  ANS1:  DEFB 'log ( ',0
B96E 282000
B971 20292069  ANS2:  DEFB ' ) is equal to ',0
B975 73206571
B979 75616C20
B97D 746F2000
          ;

```

```

a=&hb900:call a

```

```

single precision number ? 0.001
log ( 0.001 ) is equal to -6.90776
single precision number ? 0.01
log ( 0.01 ) is equal to -4.60517
single precision number ? 0.1
log ( 0.1 ) is equal to -2.30259
single precision number ? 1
log ( 1 ) is equal to 0
single precision number ? 10
log ( 10 ) is equal to 2.30259
single precision number ? 100
log ( 100 ) is equal to 4.60517
single precision number ? 10000
log ( 10000 ) is equal to 9.21034
single precision number ? 100000000
log ( 100000000 ) is equal to 18.4207
single precision number ? -100
Illegal function call
Ok

```

# 1 F 5 3 H : 単精度型実数の乗算を行なう.

アドレス	1 F 5 3 H
機 能	単精度型実数の乗算を行なう.
レジスタ	A, F, B, C, D, E, H, L

**解 説** レジスタ B C D E に格納された単精度型の実数を被乗数とし、フローティング・アキュムレータに格納された単精度型の実数を乗数として乗算を行ない、フローティング・アキュムレータに積を格納してもどります.

## サンプル

```

;
; --- multiplication for single precision ---
;
; ORG 0B900H
;
B900 215CB9 START: LD HL,PRMPT1
B903 CD53B9 CALL DSPMSG
B906 CD925F CALL 5F92H ; screen editor
B909 D8 RET C ; stop key, return to basic
B90A 23 INC HL
B90B CD41B9 CALL CLRFBAC ; clear floating point accumulator
B90E CDC826 CALL 26C8H ; convert into binary code
B911 CD1422 CALL 2214H ; convert into single precision
B914 CDE820 CALL 20E8H ; move from facc to register bcde
B917 C5 PUSH BC
B918 D5 PUSH DE
;
B919 216AB9 LD HL,PRMPT2
B91C CD53B9 CALL DSPMSG
B91F CD925F CALL 5F92H ; screen editor
B922 23 INC HL
B923 CD41B9 CALL CLRFBAC ; clear floating point accumulator
B926 CDC826 CALL 26C8H ; convert into binary code
B929 CD1422 CALL 2214H ; convert into single precision
;
B92C D1 POP DE
B92D C1 POP BC
B92E CD531F CALL 1F53H ;*multiply [reg-bcde] by [facc]
B931 CDD028 CALL 28D0H ; convert into character code
B934 E5 PUSH HL
B935 2175B9 LD HL,ANSWER
B938 CD53B9 CALL DSPMSG
B93B E1 POP HL
B93C CD53B9 CALL DSPMSG ; display result
;
B93F 18BF JR START
;
; clear floating point accumulator subroutine
;
B941 E5 CLRFBAC: PUSH HL
B942 210000 LD HL,0
B945 223DEC LD (0EC3DH),HL
B948 223FEC LD (0EC3FH),HL
B94B 2241EC LD (0EC41H),HL
B94E 2243EC LD (0EC43H),HL
B951 E1 POP HL
B952 C9 RET
;
; display message subroutine
;
B953 7E DSPMSG: LD A,(HL)

```

```

B954 A7          AND A
B955 C8          RET Z
B956 CD0D3E      CALL 3E0DH          ; display a character
B959 23          INC HL
B95A 18F7        JR DSPMSG

```

```

;
; message area
;
B95C 0D0A0A20 PRMPT1:DEFB 0DH,0AH,0AH,' x ? ',0
B960 20202078
B964 2020203F
B968 2000
B96A 20202020 PRMPT2:DEFB ' y ? ',0
B96E 79202020
B972 3F2000
B975 78202A20 ANSWER:DEFB 'x * y --> ',0
B979 79202D2D
B97D 3E2000
;

```

```

a=&hb900:call a

```

```

x ? 10000
y ? 100
x * y --> 1E+06

```

```

x ? 1
y ? 1
x * y --> 1

```

```

x ? 2
y ? 2
x * y --> 4

```

```

x ? 100
y ? 123
x * y --> 12300

```

```

x ?
Ok

```

# 1 F B 7 H : 単精度型実数の除算を行なう.

アドレス	1 F B 7 H
機能	単精度型実数の除算を行なう.
レジスタ	A, F, B, C, D, E, H, L

**解 説** レジスタ B C D E に格納された単精度型の実数を被除数とし、フローティング・アキュムレータに格納された単精度型の実数を除数として除算を行ない、フローティング・アキュムレータに商を格納してもどります.

もし、フローティング・アキュムレータに与える除数が 0 (ゼロ) であった場合には、“Division by zero”( / 0 Error) エラーとなりますがインタプリタは、“ / 0 ”のメッセージを表示し、倍精度型の実数として扱う事のできる最大の数値 ( 1, 7 0 1 4 1 1 8 3 4 6 0 4 6 9 3 D + 3 8 ) を除算の結果として与えます.

## サンプル

```

;
; --- division for single precision ---
;
      ORG 0B900H
;
START: LD  HL,PRMPT1
      CALL DSPMSG
      CALL 5F92H          ; screen editor
      RET  C              ; stop key, return to basic
      INC  HL
      CALL CLRFAC         ; clear floating point accumulator
      CALL 26C8H          ; convert into binary code
      CALL 2214H          ; convert into single precision
      CALL 20E8H          ; move from facc to register bcde
      PUSH BC
      PUSH DE
;
B919 216AB9      LD  HL,PRMPT2
B91C CD53B9      CALL DSPMSG
B91F CD925F      CALL 5F92H          ; screen editor
B922 23          INC  HL
B923 CD41B9      CALL CLRFAC         ; clear floating point accumulator
B926 CDC826      CALL 26C8H          ; convert into binary code
B929 CD1422      CALL 2214H          ; convert into single precision
;
B92C D1          POP  DE
B92D C1          POP  BC
B92E CDB71F      CALL 1FB7H          ; *divide [register bcde] by [facc]
B931 CDD028      CALL 28D0H          ; convert into character code
B934 E5          PUSH HL
B935 2175B9      LD  HL,ANSWER
B938 CD53B9      CALL DSPMSG
B93B E1          POP  HL
B93C CD53B9      CALL DSPMSG          ; display result
;
B93F 18BF      JR   START
;
; clear floating point accumulator subroutine
;
B941 E5          CLRAC:PUSH HL
B942 210000      LD  HL,0

```

```

B945 223DEC      LD      (0EC3DH),HL
B948 223FEC      LD      (0EC3FH),HL
B94B 2241EC      LD      (0EC41H),HL
B94E 2243EC      LD      (0EC43H),HL
B951 E1          POP     HL
B952 C9          RET

;
; display message subroutine
;
B953 7E          DSPMSG:LD      A,(HL)
B954 A7          AND      A
B955 C8          RET      Z
B956 CD0D3E      CALL     3E0DH      ; display a character
B959 23          INC      HL
B95A 18F7        JR       DSPMSG

;
; message area
;
B95C 0D0A0A20    PRMPT1:DEFB 0DH,0AH,0AH,' x ? ',0
B960 20202078
B964 2020203F
B968 2000
B96A 20202020    PRMPT2:DEFB ' y ? ',0
B96E 79202020
B972 3F2000
B975 78202F20    ANSWER:DEFB 'x / y --> ',0
B979 79202D20
B97D 3E2000
;

```

a=&hb900:call a

```

x ? 100000
y ? 10
x / y --> 10000

x ? 10
y ? -3
x / y --> -3.33333

x ? 5
y ? abc
/0
x / y --> 1.70141E+38

x ? 20
y ? 6
x / y --> 3.33333

x ?
Ok

```

## 20ABH: FACCの符号を反転する。

アドレス	20ABH
機能	フローティング・アキュムレータ (FACC) の符号を反転する。
レジスタ	A, F, H, L

**解説** フローティング・アキュムレータ (FACC) に格納されている、単精度型実数または倍精度型実数の符号を反転 (NEGATION) してもどります。

具体的には、フローティング・アキュムレータの仮数部最上位バイト (EC43H番地) の最上位ビット (MSB) を反転しているのみで、EABDH番地のチェック等エラー処理は全く行なっていませんから、フローティング・アキュムレータのデータは必ず実数型でなければいけません。

### サンプル

```

;
; --- negation for the real number ---
;
        ORG 0B900H
;
B900 2147B9  START: LD  HL,PROMPT
B903 CD3EB9      CALL DSPMSG
B906 CD925F      CALL 5F92H          ; screen editor
B909 D8          RET C          ; stop key, return to basic
B90A 23          INC HL
B90B CD2CB9      CALL CLRFBAC      ; clear floating point accumulator
B90E CDC826      CALL 26C8H      ; convert into binary code
;
B911 3ABDEA      LD  A,(0EABDH)    ;
B914 FE02        CP  2            ; convert into the real number
B916 CC2F22      CALL Z,222FH      ;
;
B919 CDAB20      CALL 20ABH        ;*negation for the real number
;
B91C CDD028      CALL 28D0H        ; convert into character code
B91F E5          PUSH HL
B920 215FB9      LD  HL,ANSWER
B923 CD3EB9      CALL DSPMSG
B926 E1          POP  HL
B927 CD3EB9      CALL DSPMSG      ; display result
;
B92A 18D4        JR  START
;
; clear floating point accumulator subroutine
;
B92C E5          CLRFBAC: PUSH HL
B92D 210000      LD  HL,0
B930 223DEC      LD  (0EC3DH),HL
B933 223FEC      LD  (0EC3FH),HL
B936 2241EC      LD  (0EC41H),HL
B939 2243EC      LD  (0EC43H),HL
B93C E1          POP  HL
B93D C9          RET
;
; display message subroutine
;
B93E 7E          DSPMSG: LD  A,(HL)
B93F A7          AND  A
B940 C8          RET  Z

```

```

B941 C0D03E      CALL 3E0DH      ; display a character
B944 23          INC HL
B945 18F7         JR   DSPMSG

```

```

;
; message area
;

```

```

B947 0D0A0A61    PROMPT:DEFB 0DH,0AH,0AH,'a real number x ? ',0
B94B 20726561
B94F 6C206E75
B953 6D626572
B957 20782020
B95B 203F2000
B95F 20202020    ANSWER:DEFB '          -x --> ',0
B963 20202020
B967 20202020
B96B 202D7820
B96F 202D3E20
B973 00
;

```

```

a=&hb900:call a

```

```

a real number x ? 123456789
-x --> -123456789

```

```

a real number x ? 1.23456789
-x --> -1.23456789

```

```

a real number x ? -30000
-x --> 30000

```

```

a real number x ? -5.55
-x --> 5.55

```

```

a real number x ? 398724873784279879837983729874
-x --> -3.9872487378428D+29

```

```

a real number x ?
Ok

```



## 20E8H：単精度型実数の移動を行なう。

アドレス	20E8H
機能	単精度型実数の移動を行なう。
レジスタ	B, C, D, E, H, L

**解 説** フローティング・アキュムラ (FACC) に格納されている単精度型の実数を、レジスタBCDEに移します。

ただし、次のように、メモリ上では最上位バイトのデータがレジスタBに、最下位バイトのデータがレジスタEに格納されます。

レジスタE ← EC41H番地  
 レジスタD ← EC42H番地  
 レジスタC ← EC43H番地  
 レジスタB ← EC44H番地

### サンプル

```

;
; --- dump single precision number on memory ---
;
        ORG 0B900H
;
B900 216AB9  START: LD  HL,PROMPT
B903 CD41B9  CALL DSPMSG
B906 CD925F  CALL 5F92H
B909 08      RET C
B90A 23      INC HL
B90B CD2FB9  CALL CLRAC
B90E CDC826  CALL 26C8H
B911 CD1422  CALL 2214H
B914 CDE820  CALL 20E8H
;
B917 2188B9  LD HL,ANSWER
B91A CD41B9  CALL DSPMSG
;
B91D 7B      LD A,E
B91E CD4AB9  CALL DSPACC
B921 7A      LD A,D
B922 CD4AB9  CALL DSPACC
B925 79      LD A,C
B926 CD4AB9  CALL DSPACC
B929 78      LD A,B
B92A CD4AB9  CALL DSPACC
;
B92D 18D1    JR START
;
; clear floating point accumulator subroutine
;
B92F E5      CLRFAC:PUSH HL
B930 210000  LD HL,0
B933 223DEC  LD (0EC3DH),HL
B936 223FEC  LD (0EC3FH),HL
B939 2241EC  LD (0EC41H),HL
B93C 2243EC  LD (0EC43H),HL
B93F E1      POP HL
B940 C9      RET
    
```

```

;
; display message subroutine
;
B941 7E   DSPMSG:LD   A,(HL)
B942 A7       AND   A
B943 C8       RET    Z
B944 CD0D3E   CALL  3E0DH           ; display a character
B947 23       INC   HL
B948 18F7     JR    DSPMSG

;
; display accumulator hexa decimal subroutine
;
;   destroys : register a,f,l
;
B94A 6F   DSPACC:LD   L,A
B94B 0F       RRCA
B94C 0F       RRCA
B94D 0F       RRCA
B94E 0F       RRCA
B94F CD5CB9   CALL  HALF           ; display msb four bits
B952 7D       LD    A,L
B953 CD5CB9   CALL  HALF           ; display lsb four bits
B956 3E20     LD    A,' '
B958 CD0D3E   CALL  3E0DH           ; display space code
B95B C9       RET

B95C E60F     HALF: AND  0FH
B95E FE0A     CP    10
B960 3802     JR    C,NUMBER
B962 C607     ADD   A,7

;
B964 C630     NUMBER:ADD A,30H
B966 CD0D3E   CALL  3E0DH           ; display a character
B969 C9       RET

;
; message area
;
B96A 0D0A0A73 PROMPT:DEFB 0DH,0AH,0AH,'single precision number ? ',0
B96E 696E676C
B972 65207072
B976 65636973
B97A 696F6E20
B97E 6E756D62
B982 6572203F
B986 2000
B988 20202020 ANSWER:DEFB '
B98C 20202020
B990 20202020
B994 20206F6E
B998 206D656D
B99C 6F727920
B9A0 3A2000
;

```

a=&hb900:call a

single precision number ? 0  
on memory : 00 00 00 00

single precision number ? 1  
on memory : 00 00 00 81

single precision number ? 2  
on memory : 00 00 00 82

single precision number ? 10  
on memory : 00 00 20 84

single precision number ? 100  
on memory : 00 00 48 87

single precision number ? 1000  
on memory : 00 00 7A 8A

single precision number ?  
Ok

## 2 1 3 4 H：単精度型実数の比較を行なう。

アドレス	2 1 3 4 H
機能	単精度型実数の比較を行なう。
レジスタ	A, F, H, L

**解 説** レジスタ B C D E に格納された単精度型の実数を第 1 パラメータとし、フローティング・アキュムレータ (F A C C) に格納された単精度型の実数を第 2 パラメータとして両者の比較を行ない、比較結果をアキュムレータに格納してもどります。

比較結果とアキュムレータに与える数値との関係は次のとおりです。

第 1 パラメータ < 第 2 パラメータの場合アキュムレータに 0 1 H を与える。

第 1 パラメータ = 第 2 パラメータの場合アキュムレータに 0 0 H を与える。

第 1 パラメータ > 第 2 パラメータの場合アキュムレータに F F H を与える。

### サンプル

```

;
; --- compare for single precision ---
;
; ORG 0B900H
;
B900 2164B9 ; START: LD HL,PRMPT1
B903 CD5BB9 CALL DSPMSG
B906 CD925F CALL 5F92H ; screen editor
B909 D8 RET C ; stop key, return to basic
B90A 23 INC HL
B90B CD49B9 CALL CLR FAC ; clear floating point accumulator
B90E CDC826 CALL 26C8H ; convert into binary code
B911 CD1422 CALL 2214H ; convert into single precision
B914 CDE820 CALL 20E8H ; move from facc to register bcde
B917 C5 PUSH BC
B918 D5 PUSH DE
;
B919 2160B9 ; LD HL,PRMPT2
B91C CD5BB9 CALL DSPMSG
B91F CD925F CALL 5F92H ; screen editor
B922 23 INC HL
B923 CD49B9 CALL CLR FAC ; clear floating point accumulator
B926 CDC826 CALL 26C8H ; convert into binary code
B929 CD1422 CALL 2214H ; convert into single precision
;
B92C D1 POP DE
B92D C1 POP BC
;
B92E CD3421 ; CALL 2134H ;*compare for single precision
;
B931 A7 AND A
B932 2005 JR NZ,NEQUAL
B934 2173B9 LD HL,ANS1 ; accumulator equal to 0, x!=y!
B937 180B JR DSPLAY
;
B939 F241B9 ; NEQUAL:JP P,LESS
B93C 217BB9 LD HL,ANS2 ; accumulator equal to -1, x!>y!
B93F 1803 JR DSPLAY
;
B941 2183B9 ; LESS: LD HL,ANS3 ; accumulator equal to 1, x!<y!

```

```

;
B944 CD5BB9  DISPLAY:CALL DSPMSG          ; display result
B947 18B7      JR      START

;
; clear floating point accumulator subroutine
;
B949 E5        CLRFAC:PUSH HL
B94A 210000     LD      HL,0
B94D 223DEC     LD      (0EC3DH),HL
B950 223FEC     LD      (0EC3FH),HL
B953 2241EC     LD      (0EC41H),HL
B956 2243EC     LD      (0EC43H),HL
B959 E1        POP     HL
B95A C9        RET

;
; display message subroutine
;
B95B 7E        DSPMSG:LD      A,(HL)
B95C A7        AND      A
B95D C8        RET      Z
B95E CD0D3E     CALL 3E0DH          ; display a character
B961 23        INC      HL
B962 18F7      JR      DSPMSG

;
; message area
;
B964 0D0A0A78  PRMPT1:DEFB 0DH,0AH,0AH,'x! ? ',0
B968 21203F20
B96C 00
B96D 7921203F  PRMPT2:DEFB 'y! ? ',0
B971 2000
B973 7821203D  ANS1:  DEFB 'x! = y! ',0
B977 20792100
B97B 7821203E  ANS2:  DEFB 'x! > y! ',0
B97F 20792100
B983 7821203C  ANS3:  DEFB 'x! < y! ',0
B987 20792100
;

```

a=&hb900:call a

```

x! ? 123456.789
y! ? 1234567.89
x! < y!

```

```

x! ? 123456789
y! ? 123456788
x! = y!

```

```

x! ? 1.11111
y! ? 1.11111000000001
x! = y!

```

```

x! ? 98
y! ? -98
x! > y!

```

```

x! ?
Ok

```

## 2 1 5 F H : 整数の比較を行なう.

アドレス	2 1 5 F H
機 能	整数の比較を行なう.
レジスタ	A, F

**解 説** レジスタ D E に格納された符号付き 16 ビットの整数を第 1 パラメータとし、レジスタ H L に格納された符号付き 16 ビットの整数を第 2 パラメータとして両者の比較を行ない、比較結果をアキュムレータに格納してもどります。比較結果とアキュムレータに与える数値との関係は次のとおりです。

第 1 パラメータ < 第 2 パラメータの場合アキュムレータに 0 1 H を与える。

第 1 パラメータ = 第 2 パラメータの場合アキュムレータに 0 0 H を与える。

第 1 パラメータ > 第 2 パラメータの場合アキュムレータに F F H を与える。

### サンプル

```

; --- compare for integer ---
;
; ORG 0B900H
;
B900 2166B9 START: LD HL,PRMPT1
B903 CD50B9 CALL DSPMSG
B906 CD925F CALL 5F92H ; screen editor
B909 08 RET C ; stop key, return to basic
B90A 23 INC HL
B90B CD48B9 CALL CLRFBAC ; clear floating point accumulator
B90E CDC826 CALL 26C8H ; convert into binary code
B911 CDA021 CALL 21A0H ; convert into integer
B914 ED5B41EC LD DE,(0EC41H) ; store 1st argument into reg-de
B918 D5 PUSH DE
;
B919 216FB9 LD HL,PRMPT2
B91C CD50B9 CALL DSPMSG
B91F CD925F CALL 5F92H ; screen editor
B922 23 INC HL
B923 CD48B9 CALL CLRFBAC ; clear floating point accumulator
B926 CDC826 CALL 26C8H ; convert into binary code
B929 CDA021 CALL 21A0H ; convert into integer
B92C 2A41EC LD HL,(0EC41H) ; set 2nd argument into reg-hl
;
B92F D1 POP DE ; set 1st argument into reg-de
;
B930 CD5F21 CALL 215FH ;*compare for integer
;
B933 A7 AND A
B934 2005 JR NZ,NEQUAL
B936 2175B9 LD HL,ANS1 ; accumulator equal to 0, x%=y%
B939 180B JR DISPLAY
;
B93B F243B9 NEQUAL:JP P,LESS
B93E 2170B9 LD HL,ANS2 ; accumulator equal to -1, x%>y%
B941 1803 JR DISPLAY
;
B943 2105B9 LESS: LD HL,ANS3 ; accumulator equal to 1, x%<y%
;
B946 CD50B9 DISPLAY:CALL DSPMSG ; display result
B949 18B5 JR START

```

```

;
; clear floating point accumulator subroutine
;
B94B E5      CLRFAC: PUSH HL
B94C 210000      LD HL, 0
B94F 223DEC      LD (0EC3DH), HL
B952 223FEC      LD (0EC3FH), HL
B955 2241EC      LD (0EC41H), HL
B958 2243EC      LD (0EC43H), HL
B95B E1          POP HL
B95C C9          RET

;
; display message subroutine
;
B95D 7E      DSPMSG: LD A, (HL)
B95E A7          AND A
B95F C8          RET Z
B960 C00D3E      CALL 3E0DH          ; display a character
B963 23          INC HL
B964 18F7      JR DSPMSG

;
; message area
;
B966 0D0A0A7B PRMPT1: DEFB 0DH, 0AH, 0AH, 'x% ? ', 0
B96A 25203F20
B96E 00
B96F 7925203F PRMPT2: DEFB 'y% ? ', 0
B973 2000
B975 7825203D ANS1: DEFB 'x% = y%', 0
B979 20792500
B97D 7825203E ANS2: DEFB 'x% > y%', 0
B981 20792500
B985 7825203C ANS3: DEFB 'x% < y%', 0
B989 20792500
;

```

```

a=&hb900:call a

```

```

x% ? 5
y% ? -5
x% > y%      {

x% ? 5000
y% ? 5000.4
x% = y%

```

```

x% ? 5000
y% ? 5000.5
x% < y%

```

```

x% ? 35000
Overflow
Ok

```

## 21A0H: FACCのデータを整数に変換する.

アドレス	21A0H
機能	FACCのデータを整数に変換する.
レジスタ	A, F, B, C, D, E, H, L

**解 説** フローティング・アキュムレータ (FACC) に格納された数値データを整数に変換し、レジスタHLにも格納してもどります.

フローティング・アキュムレータがすでに整数型であった場合には何もせずにもどりますが、文字型 (EABDH番地が03H) であった場合には “Type mismatch” (TM Error) エラーが起こり、データが整数型で表現できる範囲内 (−32768〜±32767) を越える場合には “Overflow” (OV Error) エラーが起こります.

具体的には、リスタートで0030H番地をコールしてフローティング・アキュムレータ (FACC) の型を調べ、次のように分岐しています.

FACCが整数型の場合、レジスタHLに移動するのみでもどります.

FACCが文字型の場合、03B1H番地 (エラー処理) に分岐します.

FACCが単精度実数型の場合、21BCH番地に分岐します.

FACCが倍精度実数型の場合、21ABH番地にエントリします.



# サンプル

```

;
; --- convert facc into integer ---
;
;       ORG 0B900H
;
B900 212AB9 ;START: LD HL,PROMPT
B903 CD21B9 CALL DSPMSG
B906 CD925F CALL 5F92H ; screen editor
B909 D8 RET C ; stop key, return to basic
B90A 23 INC HL
B90B CDBC26 CALL 26BCH ; convert into binary code
;
B90E CDA021 CALL 21A0H ;*convert facc into an integer
B911 CDD028 CALL 28D0H ; convert facc into character code
;
B914 E5 PUSH HL
B915 2139B9 LD HL,ANSWER
B918 CD21B9 CALL DSPMSG
B91B E1 POP HL
;
B91C CD21B9 CALL DSPMSG ; display an integer
B91F 18DF JR START
;
; display message subroutine
;
B921 7E DSPMSG:LD A,(HL)
B922 A7 AND A
B923 C8 RET Z
B924 CD0D3E CALL 3E0DH ; display a character
B927 23 INC HL
B928 18F7 JR DSPMSG
;
; message area
;
B92A 0D0A0A61 PROMPT:DEFB 0DH,0AH,0AH,'a number ? ',0
B92E 206E756D
B932 62657220
B936 3F2000
B939 616E2069 ANSWER:DEFB 'an integer : ',0
B93D 6E746567
B941 6572203A
B945 2000
;

```

a=&hb900:call a

a number ? 1  
an integer : 1

a number ? 1.4  
an integer : 1

a number ? 1.5  
an integer : 2

a number ? 1.6  
an integer : 2

a number ? 3.333333  
an integer : 3

a number ? -32000  
an integer : -32000

a number ?  
Ok

## 21BCH：単精度型実数を整数に変換する。

**アドレス** 21BCH

**機能** 単精度型実数を整数に変換する。

**レジスタ** A, F, B, C, D, E, H, L

**解説** フローティング・アキュムレータ (FACC) に格納した単精度型実数を整数に変換してもとります。

ただし、データが整数型で表現できる範囲内 ( $-32768 \sim \pm 32767$ ) を越える場合には “Over flow” (OV Error) エラーが起こります。

また、フローティング・アキュムレータ (FACC) の型をチェックしていないため、データは必ず単精度型実数でなくてはけません。

### サンプル

```

; --- convert single precision number into integer ---
;
; ORG 0B900H
;
B900 2139B9 START: LD HL,PROMPT
B903 CD30B9 CALL DSPMSG
B906 CD925F CALL 5F92H ; screen editor
B909 D8 RET C ; stop key, return to basic
B90A 23 INC HL
B90B CDBC26 CALL 26BCH ; convert into binary code
;
B90E 3ABDEA LD A,(0EABDH)
B911 FE04 CP 04H ; not single precision, error !!
B913 2013 JR NZ,ERROR
;
B915 CDBC21 CALL 21BCH ;single precision no. to integer
B918 CDD028 CALL 28D0H ; convert facc into character code
;
B91B E5 PUSH HL
B91C 2157B9 LD HL,ANSWER
B91F CD30B9 CALL DSPMSG
B922 E1 POP HL
;
B923 CD30B9 CALL DSPMSG ; display an integer
B926 1808 JR START
;
B928 2162B9 ERROR: LD HL,MSGERR
B92B CD30B9 CALL DSPMSG
B92E 1800 JR START
;
; display message subroutine
;
B930 7E DSPMSG:LD A,(HL)
B931 A7 AND A
B932 C8 RET Z
B933 CD0D3E CALL 3E0DH ; display a character
B936 23 INC HL
B937 18F7 JR DSPMSG
;
; message area
;
B939 0D0A0A73 PROMPT:DEFB 0DH,0AH,0AH,'single precision number ? '.0
B93D 69E676C

```

```

B941 65207072
B945 65636973
B949 696F6E20
B94D 6E756D62
B951 6572203F
B955 2000
B957 696E7465 ANSWER:DEFB 'integer : ',0
B95B 67657220
B95F 3A2000
B962 6E6F7420 MSGERR:DEFB 'not single precision !!',07H,0
B966 73696E67
B96A 6C652070
B96E 72656369
B972 73696F6E
B976 20212107
B97A 00
;

```

```

a=&hb900:call a

```

```

single precision number ? 3
not single precision !!

```

```

single precision number ? 3!
integer : 3

```

```

single precision number ? 3#
not single precision !!

```

```

single precision number ? -5000
not single precision !!

```

```

single precision number ? -5000!
integer : -5000

```

```

single precision number ? -50000
Overflow
Ok

```

## 21FDH：整数をFACCに格納する。

アドレス	21FDH
機能	整数をFACCに格納する。
レジスタ	A

**解 説** レジスタHLの符号付き16ビット整数を、整数型のフローティング・アキュムレータ（EC41H～EC42H番地）に格納し、EABDH番地にも整数型を表わす02Hを入れてもどります。

### サンプル

```

;
; --- numbers with a byte ---
;
        ORG 0B900H
;
B900 3E00        LD  A,0
;
B902 F5          LOOP: PUSH AF
B903 6F          LD  L,A
B904 CD2DB9      CALL DSPACC          ; display the integer hexa decimal
B907 2600        LD  H,0
B909 CDFD21      CALL 21FDH          ;*store register hl into facc
;
B90C 3E80        LD  A,80H          ;
B90E 0604        LD  B,4            ; appoint format
B910 0E00        LD  C,0            ;
;
B912 CD0128      CALL 28D1H          ; convert into character code
B915 CD44B9      CALL DSPMSG         ; display the integer decimal
;
B918 3E20        LD  A,' '          ;
B91A CD0D3E      CALL 3E0DH          ;
B91D CD0D3E      CALL 3E0DH          ; display four spaces
B920 CD0D3E      CALL 3E0DH          ;
B923 CD0D3E      CALL 3E0DH          ;
;
B926 F1          POP  AF
B927 3C          INC  A
B928 FE00        CP   0
B92A 20D6        JR   NZ,LOOP
;
B92C C9          RET                ; return to basic
;
; display accumulator hexa decimal subroutine
;
; destroys : register a,f,b
;
B92D 47          DSPACC:LD  B,A
B92E 0F          RRCA
B92F 0F          RRCA
B930 0F          RRCA
B931 0F          RRCA
B932 CD36B9      CALL HALF
B935 78          LD  A,B
;
B936 E60F        HALF: AND 0FH
B938 FE0A        CP   10
B93A 3802        JR   C,NUMBER
B93C C607        ADD  A,7
;
B93E C630        NUMBER:ADD A,30H
B940 CD0D3E      CALL 3E0DH          ; display a character

```

B943 C9

RET

```
;
; display message subroutine
;
```

B944 7E

DSPMSG:LD A,(HL)

B945 A7

AND A

B946 C8

RET Z

B947 CD0D3E

CALL 3E0DH

; display a character

B94A 23

INC HL

B94B 18F7

JR DSPMSG

a=&hb900:call a

00	0	01	1	02	2	03	3	04	4	05	5	06	6	07	7
08	8	09	9	0A	10	0B	11	0C	12	0D	13	0E	14	0F	15
10	16	11	17	12	18	13	19	14	20	15	21	16	22	17	23
18	24	19	25	1A	26	1B	27	1C	28	1D	29	1E	30	1F	31
20	32	21	33	22	34	23	35	24	36	25	37	26	38	27	39
28	40	29	41	2A	42	2B	43	2C	44	2D	45	2E	46	2F	47
30	48	31	49	32	50	33	51	34	52	35	53	36	54	37	55
38	56	39	57	3A	58	3B	59	3C	60	3D	61	3E	62	3F	63
40	64	41	65	42	66	43	67	44	68	45	69	46	70	47	71
48	72	49	73	4A	74	4B	75	4C	76	4D	77	4E	78	4F	79
50	80	51	81	52	82	53	83	54	84	55	85	56	86	57	87
58	88	59	89	5A	90	5B	91	5C	92	5D	93	5E	94	5F	95
60	96	61	97	62	98	63	99	64	100	65	101	66	102	67	103
68	104	69	105	6A	106	6B	107	6C	108	6D	109	6E	110	6F	111
70	112	71	113	72	114	73	115	74	116	75	117	76	118	77	119
78	120	79	121	7A	122	7B	123	7C	124	7D	125	7E	126	7F	127
80	128	81	129	82	130	83	131	84	132	85	133	86	134	87	135
88	136	89	137	8A	138	8B	139	8C	140	8D	141	8E	142	8F	143
90	144	91	145	92	146	93	147	94	148	95	149	96	150	97	151
98	152	99	153	9A	154	9B	155	9C	156	9D	157	9E	158	9F	159
A0	160	A1	161	A2	162	A3	163	A4	164	A5	165	A6	166	A7	167
A8	168	A9	169	AA	170	AB	171	AC	172	AD	173	AE	174	AF	175
B0	176	B1	177	B2	178	B3	179	B4	180	B5	181	B6	182	B7	183
B8	184	B9	185	BA	186	BB	187	BC	188	BD	189	BE	190	BF	191
C0	192	C1	193	C2	194	C3	195	C4	196	C5	197	C6	198	C7	199
C8	200	C9	201	CA	202	CB	203	CC	204	CD	205	CE	206	CF	207
D0	208	D1	209	D2	210	D3	211	D4	212	D5	213	D6	214	D7	215
D8	216	D9	217	DA	218	DB	219	DC	220	DD	221	DE	222	DF	223
E0	224	E1	225	E2	226	E3	227	E4	228	E5	229	E6	230	E7	231
E8	232	E9	233	EA	234	EB	235	EC	236	ED	237	EE	238	EF	239
F0	240	F1	241	F2	242	F3	243	F4	244	F5	245	F6	246	F7	247
F8	248	F9	249	FA	250	FB	251	FC	252	FD	253	FE	254	FF	255

Ok

## 2202H: FACCの型を指定する.

アドレス	2202H
機能	FACCの型を指定する.
レジスタ	全て保存

**解説** アキュムレータの数値をフローティング・アキュムレータの型として, EABDH番地に格納してもどります.

### サンプル

```

;
; --- convert on memory code into the number ---
;
        ORG 0B900H
;
B900 1171B9   START: LD  DE,PRMPT1
B903 CD68B9   CALL  DSPMSG
;
B906 CD8335   CALL  3583H           ; input a character from keyboard
B909 CD0D3E   CALL  3E0DH           ; echo-back a character
B90C FE03     CP      03H
B90E C8       RET  Z               ; stop key, return to basic
B90F D630     SUB  '0'             ; convert into binary code
B911 CD0222   CALL  2202H           ; *set type of floating accumulator
B914 47       LD      B,A          ; set loop counter for input
;
B915 2141EC   LD      HL,0EC41H    ; top address for single precision
;
B918 FE08     CP      8
B91A 2005     JR      NZ,NEXTIN
B91C 213DEC   LD      HL,0EC3DH    ; top address for double precision
B91F 1800     JR      NEXTIN
;
B921 118AB9   NEXTIN:LD  DE,PRMPT2
B924 CD68B9   CALL  DSPMSG
;
B927 CD8335   CALL  3583H           ; input a character from keyboard
B92A CD0D3E   CALL  3E0DH           ; echo-back a character
B92D CD5CB9   CALL  HEXBIN         ; convert hexa code into binary
B930 4F       LD      C,A
;
B931 CD8335   CALL  3583H           ; input a character from keyboard
B934 CD0D3E   CALL  3E0DH           ; echo-back a character
B937 FE0D     CP      0DH
B939 280D     JR      Z,CRCODE
B93B CD5CB9   CALL  HEXBIN         ; convert hexa code into binary
;
B93E CB01     RLC  C
B940 CB01     RLC  C
B942 CB01     RLC  C
B944 CB01     RLC  C
B946 81       ADD  A,C
B947 4F       LD      C,A
;
B948 71       CRCODE:LD  (HL),C
B949 23       INC  HL
B94A 10D5     DJNZ NEXTIN
;
B94C CD0D28   CALL  28D0H           ; convert into character code
B94F E5       PUSH HL              ; push top address onto stack

```

```

;
B950 11A2B9      LD  DE,ANSWER
B953 CD68B9      CALL DSPMSG

;
B956 D1          POP  DE                      ; pop top address from stack
B957 CD68B9      CALL DSPMSG                  ; display the number

;
B95A 18A4        JR   START

;
; convert hexa code into binary code subroutine
;
;   inputs   : character code in accumulator
;   outputs  : binary code in accumulator
;   destroys : register a,f
;
B95C FE3A        HEXBIN:CP  '9'+1
B95E 3805        JR   C,HEXB11
B960 E60F        AND  0FH
B962 C609        ADD  A,9
B964 C9          RET

;
B965 E60F        HEXB11:AND  0FH
B967 C9          RET

;
; display message subroutine
;
;   destroys : register a,f,d,e
;
B968 1A          DSPMSG:LD  A,(DE)
B969 A7          AND  A
B96A C8          RET  Z
B96B CD0D3E      CALL 3E0DH
B96E 13          INC  DE
B96F 18F7        JR   DSPMSG

;
; data area
;
B971 0D0A0A74    PRMPT1:DEFB 0DH,0AH,0AH,'type of the number ? ',0
B975 79706520
B979 6F662074
B97D 6865206E
B981 756D6265
B985 72203F20
B989 00
B98A 0D0A2020    PRMPT2:DEFB 0DH,0AH,      '          on memory ? ',0
B98E 20202020
B992 2020206F
B996 6E206D65
B99A 6D6F7279
B99E 203F2000
B9A2 0D0A2020    ANSWER:DEFB 0DH,0AH,      '          the number : ',0
B9A6 20202020
B9AA 20207468
B9AE 65206E75
B9B2 6D626572
B9B6 203A2000
;

```



a=&hb900:call a

type of the number ? 2  
on memory ? 00  
on memory ? 00  
the number : 0

type of the number ? 2  
on memory ? 01  
on memory ? 00  
the number : 1

type of the number ? 2  
on memory ? 50  
on memory ? 00  
the number : 80

type of the number ? 2  
on memory ? ff  
on memory ? ff  
the number : -1

type of the number ? 4  
on memory ? 11  
on memory ? 11  
on memory ? 11  
on memory ? 11  
the number : 2.18272E-34

type of the number ? 8  
on memory ? 00  
on memory ? 00  
on memory ? 00  
on memory ? 00  
on memory ? 00  
on memory ? 00  
on memory ? 00  
on memory ? 84  
the number : 8

type of the number ?  
Ok

## フローティング・アキュムレータの構成

N<sub>88</sub>-BASIC インタプリタは、浮動小数点の、演算機能が用意されていない Z 8 0 (μP D 7 8 0) で浮動小数点の演算を実行するため、フローティング・アキュムレータ (FACC) と呼ばれる 8 バイトの仮想レジスタをメモリ上に用意し、全てソフトウェアによって浮動小数点の演算処理を行なっています。

このフローティング・アキュムレータ (FACC) は、E C 3 D H ~ E C 4 4 H 番地に置かれ、E A B D H 番地にはデータの型を表わす数値が入っています。以下に、それぞれの型におけるフローティング・アキュムレータの構成を示します。

### ● 整数型

E C 4 1 H	下位 8 ビット	E A B D H	0 2 H
E C 4 2 H	上位 8 ビット		

### ● 単精度実数型

E C 4 1 H	仮数部下位 8 ビット	E A B D H	0 4 H
E C 4 2 H	仮数部中位 8 ビット		
E C 4 3 H	仮数部上位 8 ビット		
E C 4 4 H	指数部 8 ビット		

### ● 倍精度実数型

E C 3 D H	仮数部最下位 8 ビット	E A B D H	0 8 H
E C 3 E H	"		
E C 3 F H	"		
E C 4 0 H	"		
E C 4 1 H	"		
E C 4 2 H	"		
E C 4 3 H	仮数部最上位 8 ビット		
E C 4 4 H	指数部 8 ビット		

## 2 2 1 4 H : F A C C のデータを単精度型に変換する.

アドレス	2 2 1 4 H
機 能	F A C C のデータを単精度型に変換する.
レジスタ	A, F, B, C, D, E, H, L

**解 説** フローティング・アキュムレータ (F A C C) に格納された数値データを単精度型実数に変換してもどります.

フローティング・アキュムレータ (F A C C) がすでに単精度実数型であった場合には何もせずにもどりますが, 文字型 (E A B D H 番地が 0 3 H) であった場合には “Type mismatch” (T M Error) エラーが起こります.

具体的には, リスタートで 0 0 3 0 H 番地をコールしてフローティング・アキュムレータ (F A C C) の型を調べ, 次のように分岐しています.

F A C C が整数型の場合, 2 2 2 F H 番地に分岐します.

F A C C が文字型の場合, 0 3 B 1 H 番地 (エラー処理) に分岐します.

F A C C が単精度実数型の場合, 何もせずにもどります.

F A C C が倍精度実数型の場合, 2 2 1 C H 番地にエントリします.

# サンプル

```

;
; --- convert facc into single precision ---
;
        ORG 0B900H
;
B900 212AB9  START: LD  HL,PROMPT
B903 CD21B9  CALL DSPMSG
B906 CD925F  CALL 5F92H          ; screen editor
B909 08      RET C              ; stop key, return to basic
B90A 23      INC HL
B90B CD8C26  CALL 26BCH          ; convert into binary code
;
B90E CD1422  CALL 2214H          ;*convert facc into single precisi
B911 CD0028  CALL 28D0H          ; convert facc into character code
;
B914 E5      PUSH HL
B915 2139B9  LD  HL,ANSWER
B918 CD21B9  CALL DSPMSG
B91B E1      POP HL
;
B91C CD21B9  CALL DSPMSG          ; display single precision number
B91F 18DF    JR  START
;
; display message subroutine
;
B921 7E      DSPMSG:LD  A,(HL)
B922 A7      AND  A
B923 C8      RET  Z
B924 CD0D3E  CALL 3E0DH          ; display a character
B927 23      INC HL
B928 18F7    JR  DSPMSG
;
; message area
;
B92A 0D0A0A61 PROMPT:DEFB 0DH,0AH,0AH,'a number ? ',0
B92E 206E756D
B932 62657220
B936 3F2000
B939 73696E67 ANSWER:DEFB 'single precision number : ',0
B93D 6C652070
B941 72656369
B945 73696F6E
B949 206E756D
B94D 62657220
B951 3A2000
;

```

a=&hb900:call a

```

a number ? 1
single precision number : 1

a number ? 1.00001
single precision number : 1.00001

a number ? 1.00000001
single precision number : 1

a number ? 3333333333333333
single precision number : 3.33333E+15

a number ? 0
single precision number : 0

a number ? 0.5
single precision number : .5

a number ?
Ok

```

## 2 2 1 C H : 倍精度型実数を単精度型実数に変換する.

**アドレス** 2 2 1 C H

**機能** 倍精度型実数を単精度型実数に変換する.

**レジスタ** A, F, B, C, D, E, H, L

**解説** フローティング・アキュムレータ (FACC) に格納されている倍精度型実数を単精度型の実数に変換してもどります.

### サンプル

```

;
; --- convert double prcision into single precision ---
;
        ORG 0B900H
;
B900 2139B9 ; START: LD HL,PROMPT
B903 CD30B9 ; CALL DSPMSG
B906 CD925F ; CALL 5F92H ; screen editor
B909 D8 ; RET C ; stop key, return to basic
B90A 23 ; INC HL
B90B CDBC26 ; CALL 26BCH ; convert into binary code
;
B90E 3ABDEA ; LD A,(0EABDH)
B911 FE08 ; CP 08H ; not double precision, error !!
B913 2013 ; JR NZ,ERROR
;
B915 CD1C22 ; CALL 221CH ;*convert from double precion no.
B918 CDD028 ; CALL 28D0H ; convert facc into character code
;
B91B E5 ; PUSH HL
B91C 2157B9 ; LD HL,ANSWER
B91F CD30B9 ; CALL DSPMSG
B922 E1 ; POP HL
;
B923 CD30B9 ; CALL DSPMSG ; display single precision number
B926 1808 ; JR START
;
B928 2172B9 ; ERROR: LD HL,MSGERR
B92B CD30B9 ; CALL DSPMSG
B92E 1800 ; JR START
;
; display message subroutine
;
B930 7E ; DSPMSG:LD A,(HL)
B931 A7 ; AND A
B932 C8 ; RET Z
B933 CD0D3E ; CALL 3E0DH ; display a character
B936 23 ; INC HL
B937 18F7 ; JR DSPMSG
;
; message area
;
B939 0D0A0A64 ; PROMPT:DEFB 0DH,0AH,0AH,'double precision number ? ',0
B93D 6F75626C
B941 65207072
B945 65636973
B949 696F6E20
B94D 6E756D62
B951 6572203F
B955 2000
B957 73696E67 ; ANSWER:DEFB 'single precision number : ',0
B95B 6C652070
B95F 72656369

```



## 2 2 2 F H : 整数を単精度型実数に変換する.

**アドレス** 2 2 2 F H

**機能** 整数を単精度型実数に変換する.

**レジスタ** A, F, B, C, D, E, H, L

**解説** フローティング・アキュムレータ (FACC) に格納されている整数を単精度型の実数に変換してもどります.

### サンプル

```

;
; --- convert integer into single precision number ---
;
; ORG 0B900H
;
B900 2139B9 START: LD HL,PROMPT
B903 CD30B9 CALL DSPMSG
B906 CD925F CALL 5F92H ; screen editor
B909 D8 RET C ; stop key, return to basic
B90A 23 INC HL
B90B CDBC26 CALL 26BCH ; convert into binary code
;
B90E 3ABDEA LD A,(0EABDH) ;
B911 FE02 CP 02H ;
B913 2013 JR NZ,ERROR ; not integer, error !!
;
B915 CD2F22 CALL 222FH ;*convert from integer
B918 CDD028 CALL 28D0H ; convert facc into character code
;
B91B E5 PUSH HL
B91C 2147B9 LD HL,ANSWER
B91F CD30B9 CALL DSPMSG
B922 E1 POP HL
;
B923 CD30B9 CALL DSPMSG ; display single precision number
B926 18D8 JR START
;
B928 2162B9 ERROR: LD HL,MSGERR
B92B CD30B9 CALL DSPMSG
B92E 18D0 JR START
;
; display message subroutine
;
B930 7E DSPMSG:LD A,(HL)
B931 A7 AND A
B932 C8 RET Z
B933 CD0D3E CALL 3E0DH ; display a character
B936 23 INC HL
B937 18F7 JR DSPMSG
;
; message area
;
B939 0D0A0A69 PROMPT:DEFB 0DH,0AH,0AH,'integer ? ',0
B93D 6E746567
B941 6572203F
B945 2000
B947 73696E67 ANSWER:DEFB 'single precision number : ',0
B94B 6C652070
B94F 72656369
B953 73696F6E
B957 206E756D
B95B 62657220
B95F 3A2000

```



```
B962 6E6F7420 MSGERR:DEFB 'not integer !!',07H,0
B966 696E7465
B96A 67657220
B96E 21210700
;
```

```
a=&hb900:call a
```

```
integer ? 0
single precision number : 0
```

```
integer ? 1
single precision number : 1
```

```
integer ? 1#
not integer !!
```

```
integer ? 1.5
not integer !!
```

```
integer ? -5000
single precision number : -5000
```

```
integer ? 33000
not integer !!
```

```
integer ?
Ok
```

## 2 2 3 E H : F A C C のデータを倍精度型実数に変換する。

アドレス	2 2 3 E H
機 能	F A C C のデータを倍精度型実数に変換する。
レジスタ	A, F, B, C, D, E, H, L

**解 説** フローティング・アキュムレータ (F A C C) に格納された数値データを倍精度型実数に変換してもどります。

フローティング・アキュムレータ (F A C C) の数値データがすでに倍精度型実数であった場合には何もせずにもどりますが、文字型データ (E A B D H 番地が 0 3 H) であった場合にはエラーが発生します。

具体的には、リスタートで 0 0 3 0 H 番地をコールし、フローティング・アキュムレータの型を調べて次のように分岐しています。

F A C C が整数型の場合 2 2 2 F H 番地に分岐します。

F A C C が文文字型の場合 0 3 B 1 H 番地に (エラー処理) に分岐します。

F A C C が単精度実数型の場合 2 2 4 6 H 番地にエントリします。

F A C C が倍精度実数型の場合何もせずにもどります。

### サンプル

```

;
; --- convert facc into double precision ---
;
; ORG 0B900H
;
B900 212AB9 START: LD HL,PROMPT
B903 CD21B9 CALL DSPMSG
B906 CD925F CALL 5F92H ; screen editor
B909 D8 RET C ; stop key, return to basic
B90A 23 INC HL
B90B CDBC26 CALL 26BCH ; convert into binary code
;
B90E CD3E22 CALL 223EH ;*convert facc into double precisi
B911 CDD028 CALL 28D0H ; convert facc into character code
;
B914 E5 PUSH HL
B915 2139B9 LD HL,ANSWER
B918 CD21B9 CALL DSPMSG
B91B E1 POP HL
;
B91C CD21B9 CALL DSPMSG ; display double precision number
B91F 18DF JR START
;
; display message subroutine
;
B921 7E DSPMSG:LD A,(HL)
B922 A7 AND A
B923 C8 RET Z
B924 CD0D3E CALL 3E0DH ; display a character
B927 23 INC HL
B928 18F7 JR DSPMSG

```

```

;
; message area
;
B92A 0D0A0A61 PROMPT:DEFB 0DH,0AH,0AH,'a number ? ',0
B92E 206E756D
B932 62657220
B936 3F2000
B939 646F7562 ANSWER:DEFB 'double precision number : ',0
B93D 6C652070
B941 72656369
B945 73696F6E
B949 206E756D
B94D 62657220
B951 3A2000
;

```

```

a=&hb900:call a

```

```

a number ? 1
double precision number : 1

```

```

a number ? 3
double precision number : 3

```

```

a number ? 5.1
double precision number : 5.099999904632568

```

```

a number ? 0.00001
double precision number : 9.99999883788405D-06

```

```

a number ?
Ok
a#=5.1:print a#
5.099999904632568
Ok
a#=0.00001:print a#
9.99999883788405D-06
Ok

```

## 2 2 4 6 H：単精度型実数を倍精度型実数に変換する。

アドレス	2 2 4 6 H
機能	単精度型実数を倍精度型実数に変換する。
レジスタ	A, B, C, H, L

**解 説** フローティング・アキュムレータに格納されている単精度型実数を倍精度型の実数に変換してもどります。

実際には、EC3DH～EC40H番地を00Hでクリアした後、EABDH番地の型を表わす数値を08Hに変更しているだけで、入力パラメータのチェック等エラーの検出は行ないません。

### サンプル

```

;
; --- convert single precision into double precision ---
;
; ORG 0B900H
;
B900 2139B9 START: LD HL,PROMPT
B903 CD30B9 CALL DSPMSG
B906 CD925F CALL 5F92H ; screen editor
B909 D8 RET C ; stop key, return to basic
B90A 23 INC HL
B90B CDBC26 CALL 26BCH ; convert into binary code
;
B90E 3ABDEA LD A,(0EABDH) ;
B911 FE04 CP 04H ; not single precision, error !!
B913 2013 JR NZ,ERROR ;
;
B915 CD4622 CALL 2246H ;*convert from single precision no.
B918 CDD028 CALL 28D0H ; convert facc into character code
;
B91B E5 PUSH HL
B91C 2157B9 LD HL,ANSWER
B91F CD30B9 CALL DSPMSG
B922 E1 POP HL
;
B923 CD30B9 CALL DSPMSG ; display double precision number
B926 18D8 JR START
;
B928 2172B9 ERROR: LD HL,MSGERR
B92B CD30B9 CALL DSPMSG
B92E 18D0 JR START
;
; display message subroutine
;
B930 7E DSPMSG:LD A,(HL)
B931 A7 AND A
B932 C8 RET Z
B933 CD0D3E CALL 3E0DH ; display a character
B936 23 INC HL
B937 18F7 JR DSPMSG
;
; message area
;
B939 0D0A0A73 PROMPT:DEFB 0DH,0AH,0AH,'single precision number ? ',0
B93D 696E676C
B941 65207072
B945 65636973

```

```

B949 696F6E20
B94D 6E756D62
B951 6572203F
B955 2000
B957 646F7562 ANSWER:DEFB 'double precision number : ',0
B95B 6C652070
B95F 72656369
B963 73696F6E
B967 206E756D
B96B 62657220
B96F 3A2000
B972 6E6F7420 MSGERR:DEFB 'not single precision !!',07H,0
B976 73696E67
B97A 6C652070
B97E 72656369
B982 73696F6E
B986 20212107
B98A 00

```

;

```
a=&hb900:call a
```

```
single precision number ? 10000
not single precision !!
```

```
single precision number ? 100000
double precision number : 100000
```

```
single precision number ? 10000000000
not single precision !!
```

```
single precision number ? 123.456
double precision number : 123.4559936523438
```

```
single precision number ? 5!
double precision number : 5
```

```
single precision number ?
```

```
Ok
a#=123.456!:print a#
123.4559936523438
Ok
```

## 2 2 5 2 H : F A C C の型を単精度実数型に指定する.

アドレス 2 2 5 2 H

機能 F A C C の型を単精度実数型に指定する.

レジスタ A

**解 説** フローティング・アキュムレータの型を表す E A B D H 番地に、単精度実数型を表す 0 4 H を入れてもどります.

具体的には、アキュムレータに 0 4 H を入れて 2 2 0 2 H 番地にジャンプしています.

### サンプル

```

;
; --- calculate r^2*pi ---
;
;          ORG 0B900H
;
B900 213EB9 START: LD HL,PROMPT
B903 CD35B9 CALL DSPMSG
B906 CD925F CALL 5F92H ; screen editor
B909 08 RET C ; stop key, return to basic
B90A 23 INC HL
B90B CDBC26 CALL 26BCH ; convert into binary code
B90E CD1422 CALL 2214H ; convert into single precision
B911 CDE820 CALL 20E8H ; move from facc to register bcde
B914 C0531F CALL 1F53H ; multiply r by r
;
B917 1EDB LD E,0DBH ;
B919 160F LD D,0FH ;
B91B 0E49 LD C,49H ; constant number of 3.14159
B91D 0682 LD B,82H ;
B91F CD5222 CALL 2252H ;
;
B922 C0531F CALL 1F53H ; multiply r^2 by pi
B925 CDD028 CALL 28D0H ; convert into character code
B928 E5 PUSH HL
B929 2148B9 LD HL,ANSWER
B92C CD35B9 CALL DSPMSG
B92F E1 POP HL
B930 CD35B9 CALL DSPMSG ; display result
;
B933 18CB JR START
;
; display message subroutine
;
B935 7E DSPMSG:LD A,(HL)
B936 A7 AND A
B937 C8 RET Z
B938 CDD03E CALL 3E0DH ; display a character
B93B 23 INC HL
B93C 18F7 JR DSPMSG
;
; message area
;
B93E 0D0A0A72 PROMPT:DEFB 0DH,0AH,0AH,'r ? ',0
B942 20202020
B946 20203F20
B94A 00
B94B 725E322A ANSWER:DEFB 'r^2*pi : ',0
B94F 7069203A
B953 2000

```

```
a=&hb900:call a
```

```
r      ? 0  
r^2*pi : 0
```

```
r      ? 0.1  
r^2*pi : .0314159
```

```
r      ? 1  
r^2*pi : 3.14159
```

```
r      ? 10  
r^2*pi : 314.159
```

```
r      ? 1.1  
r^2*pi : 3.80133
```

```
r      ? -5  
r^2*pi : 78.5398
```

```
r      ?  
Ok
```



## 2 3 2 F H : 整数の減算を行なう.

**アドレス** 2 3 2 F H

**機能** 整数の減算を行なう.

**レジスタ** A, F, B, C, D, E, H, L

**解説** レジスタDEに格納された符号付16ビット長の整数を被減数とし、レジスタHLに格納された符号付16ビット長の整数を減数として両者の間で減算を行ない、差をレジスタHLおよびフローティング・アキュムレータ (FAC C) に格納してもどります.

### サンプル

```

;
; --- subtraction for integer ---
;
        ORG 0B900H
;
B900 215EB9   START: LD  HL,PRMPT1
B903 CD55B9   CALL  DSPMSG
B906 CD925F   CALL  5F92H          ; screen editor
B909 08       RET  C              ; stop key, return to basic
B90A 23       INC  HL
B90B CD43B9   CALL  CLRFAC         ; clear floating point accumulator
B90E CDC826   CALL  26C8H         ; convert into binary code
B911 CDA021   CALL  21A0H         ; convert into integer
B914 ED5B41EC LD  DE,(0EC41H)      ; store 1st argument into reg-de
B918 D5       PUSH DE
;
B919 216EB9   LD  HL,PRMPT2
B91C CD55B9   CALL  DSPMSG
B91F CD925F   CALL  5F92H          ; screen editor
B922 23       INC  HL
B923 CD43B9   CALL  CLRFAC         ; clear floating point accumulator
B926 CDC826   CALL  26C8H         ; convert into binary code
B929 CDA021   CALL  21A0H         ; convert into integer
B92C 2A41EC   LD  HL,(0EC41H)      ; set 2nd argument into reg-hl
;
B92F D1       POP  DE              ; set 1st argument into reg-de
;
B930 CD2F23   CALL  232FH          ;*subtraction for integer
;
B933 CDD028   CALL  28D0H          ; convert into character code
B936 E5       PUSH HL
B937 2178B9   LD  HL,ANSWER
B93A CD55B9   CALL  DSPMSG
B93D E1       POP  HL
B93E CD55B9   CALL  DSPMSG          ; display result
;
B941 18BD     JR   START
;
; clear floating point accumulator subroutine
;
B943 E5       CLRFAC: PUSH HL
B944 210000   LD  HL,0
B947 223DEC   LD  (0EC3DH),HL
B94A 223FEC   LD  (0EC3FH),HL
B94D 2241EC   LD  (0EC41H),HL
B950 2243EC   LD  (0EC43H),HL
B953 E1       POP  HL
B954 C9       RET

```

```

;
; display message subroutine
;
B955 7E      DSPMSG:LD   A,(HL)
B956 A7      AND   A
B957 C8      RET    Z
B958 CD0D3E  CALL 3E0DH      ; display a character
B95B 23      INC   HL
B95C 18F7    JR     DSPMSG
;
; message area
;
B95E 0D0A0A20 PRMPT1:DEFB 0DH,0AH,0AH,'      x%  ? ',0
B962 20202020
B966 78252020
B96A 203F2000
B96E 20202020 PRMPT2:DEFB '      y%  ? ',0
B972 20792520
B976 20203F20
B97A 00
B97B 78252020 ANSWER:DEFB 'x% - y% --> ',0
B97F 20792520
B983 2D2D3E20
B987 00
;

```

a=&hb900:call a

```

      x%  ? 123
      y%  ? 23
x% - y% --> 100

      x%  ? 123
      y%  ? 0.4
x% - y% --> 123

      x%  ? 123
      y%  ? 0.4
x% - y% --> 123

      x%  ? 123
      y%  ? 0.5
x% - y% --> 122

      x%  ?
Ok

```

## 2 3 3 A H : 整数の加算を行なう.

アドレス	2 3 3 A H
機 能	整数の加算を行なう.
レジスタ	A, F, B, C, D, E, H, L

**解 説** レジスタ D E の符号付 16 ビット整数とレジスタ H L の符号付 16 ビット整数の間で加算を行ない、和をレジスタ H L およびフローティング・アキュムレータ (F A C C) に格納してもどります。

加算結果が整数型として扱え得る範囲を越えた場合には、和は単精度の値としてフローティング・アキュムレータ (F A C C) に返されますが、その場合にはレジスタ H L に与えられた和は意味を持ちません。

### サンプル

```

;
; --- addition for integer ---
;
; ORG 0B900H
B900 215EB9 START: LD HL,PRMPT1
B903 CD55B9 CALL DSPMSG
B906 CD925F CALL 5F92H ; screen editor
B909 D8 RET C ; stop key, return to basic
B90A 23 INC HL
B90B CD43B9 CALL CLRFAC ; clear floating point accumulator
B90E CDC826 CALL 26C8H ; convert into binary code
B911 CDA021 CALL 21A0H ; convert into integer
B914 ED5B41EC LD DE,(0EC41H) ; store 1st argument into reg-de
B918 D5 PUSH DE
;
B919 216EB9 LD HL,PRMPT2
B91C CD55B9 CALL DSPMSG
B91F CD925F CALL 5F92H ; screen editor
B922 23 INC HL
B923 CD43B9 CALL CLRFAC ; clear floating point accumulator
B926 CDC826 CALL 26C8H ; convert into binary code
B929 CDA021 CALL 21A0H ; convert into integer
B92C 2A41EC LD HL,(0EC41H) ; set 2nd argument into reg-hl
;
B92F D1 POP DE ; set 1st argument into reg-de
;
B930 CD3A23 CALL 233AH ;*addition for integer
;
B933 CDD028 CALL 28D0H ; convert into character code
B936 E5 PUSH HL
B937 217BB9 LD HL,ANSWER
B93A CD55B9 CALL DSPMSG
B93D E1 POP HL
B93E CD55B9 CALL DSPMSG ; display result
;
B941 18BD JR START
;
; clear floating point accumulator subroutine
;
B943 E5 CLR FAC: PUSH HL
B944 210000 LD HL,0
B947 223DEC LD (0EC3DH),HL
B94A 223FEC LD (0EC3FH),HL

```

```

B94D 2241EC      LD      (0EC41H),HL
B950 2243EC      LD      (0EC43H),HL
B953 E1          POP     HL
B954 C9          RET

;
; display message subroutine
;
B955 7E          DSPMSG:LD      A,(HL)
B956 A7          AND      A
B957 C8          RET      Z
B958 CD0D3E      CALL 3E0DH      ; display a character
B95B 23          INC      HL
B95C 18F7        JR      DSPMSG

;
; message area
;
B95E 0D0A0A20    PRMPT1:DEFB 0DH,0AH,0AH,'      x%      ? ',0
B962 20202020
B966 78252020
B96A 203F2000
B96E 20202020    PRMPT2:DEFB '      y%      ? ',0
B972 20792520
B976 20203F20
B97A 00
B97B 7825202B    ANSWER:DEFB 'x% + y% --> ',0
B97F 20792520
B983 2D2D3E20
B987 00
;

```

```

a=&hb900:call a

```

```

      x%      ? 20
      y%      ? 30
x% + y% --> 50

      x%      ? 500
      y%      ? -600
x% + y% --> -100

      x%      ? 10
      y%      ? 0.5
x% + y% --> 11

      x%      ? 300
      y%      ?
x% + y% --> 300

      x%      ?
Ok

```

## 2 3 5 A H : 整数の乗算を行なう.

**アドレス** 2 3 5 A H

**機能** 整数の乗算を行なう.

**レジスタ** A, F, B, C, D, E, H, L

**解説** レジスタDEの符号付16ビット整数とレジスタHLの符号付16ビット整数の間で乗算を行ない, 積をレジスタHLに格納してもどります.

乗算結果が整数型として扱え得る範囲を越えた場合には, 積は単精度の値としてフローティング・アキュムレータ (FACC) に返されますが, その場合にはレジスタHLに与えられた積は意味を持ちません.

### サンプル

```

; --- multiplication for integer ---
;
;      ORG 0B900H
;
B900 215EB9  START: LD  HL,PRMPT1
B903 CD55B9      CALL DSPMSG
B906 CD925F      CALL SF92H          ; screen editor
B909 D8          RET C              ; stop key, return to basic
B90A 23          INC HL
B90B CD43B9      CALL CLRFAC        ; clear floating point accumulator
B90E CDC826      CALL 26C8H        ; convert into binary code
B911 CDA021      CALL 21A0H        ; convert into integer
B914 ED5B41EC    LD  DE,(0EC41H)    ; store 1st argument into reg-de
B918 D5          PUSH DE
;
B919 216EB9      LD  HL,PRMPT2
B91C CD55B9      CALL DSPMSG
B91F CD925F      CALL SF92H          ; screen editor
B922 23          INC HL
B923 CD43B9      CALL CLRFAC        ; clear floating point accumulator
B926 CDC826      CALL 26C8H        ; convert into binary code
B929 CDA021      CALL 21A0H        ; convert into integer
B92C 2A41EC      LD  HL,(0EC41H)    ; set 2nd argument into reg-hl
;
B92F D1          POP  DE              ; set 1st argument into reg-de
;
B930 CD5A23      CALL 235AH          ;*multiplication for integer
;
B933 CDD028      CALL 28D0H          ; convert into character code
B936 E5          PUSH HL
B937 217BB9      LD  HL,ANSWER
B93A CD55B9      CALL DSPMSG
B93D E1          POP  HL
B93E CD55B9      CALL DSPMSG          ; display result
;
B941 18BD        JR   START
;
; clear floating point accumulator subroutine
;
B943 E5          CLRFAC:PUSH HL
B944 210000      LD  HL,0
B947 223DEC      LD  (0EC3DH),HL
B94A 223FEC      LD  (0EC3FH),HL
B94D 2241EC      LD  (0EC41H),HL
B950 2243EC      LD  (0EC43H),HL
B953 E1          POP  HL

```

```

B954 C9          RET
                ;
                ; display message subroutine
                ;
B955 7E          DSPMSG:LD  A,(HL)
B956 A7          AND  A
B957 C8          RET  Z
B958 CD0D3E      CALL 3E0DH          ; display a character
B95B 23          INC  HL
B95C 18F7        JR   DSPMSG
                ;
                ; message area
B95E 0D0A0A20    PRMPT1:DEFB 0DH,0AH,0AH,'    x%  ? ',0
B962 20202020
B966 78252020
B96A 203F2000
B96E 20202020    PRMPT2:DEFB '        y%  ? ',0
B972 20792520
B976 20203F20
B97A 00
B97B 7825202A    ANSWER:DEFB 'x% * y% --> ',0
B97F 20792520
B983 2D2D3E20
B987 00
                ;

```

a=&hb900:call a

```

        x%  ? 1000
        y%  ? 123
x% * y% --> 123000

        x%  ? 123.456
        y%  ? 789
x% * y% --> 97047

        x%  ? -500
        y%  ? 50
x% * y% --> -25000

        x%  ? 1000
        y%  ? 60000
Overflow
Ok

```

## 2 3 A B H : 整数の除算を行なう.

アドレス	2 3 A B H
機能	整数の除算を行なう.
レジスタ	A, F, B, C, D, E, H, L

**解 説** レジスタDEの符号付16ビット整数とレジスタHLの符号付16ビット整数の間で除算を行ない、商をレジスタHLに、乗余をレジスタDEに、それぞれ格納してもどります。

ただし、レジスタHLに与える除数が、0000H（ゼロ）であった場合にのみ、“Division by zero”(／0 Error) エラーが発生するためN88-BASICまたは機械語モニタの制御下に入ります。

### サンプル

```

;
; --- division for integer ---
;
;       ORG  0B900H
;
B900 215EB9  START: LD   HL,PRMPT1
B903 CD55B9  CALL  DSPMSG
B906 CD925F  CALL  5F92H          ; screen editor
B909 D8      RET  C          ; stop key, return to basic
B90A 23      INC  HL
B90B CD43B9  CALL  CLRFAC      ; clear floating point accumulator
B90E CDC826  CALL  26C8H      ; convert into binary code
B911 CDA021  CALL  21A0H      ; convert into integer
B914 ED5B41EC LD  DE,(0EC41H)      ; store 1st argument into reg-de
B918 D5      PUSH DE
;
B919 216EB9  LD   HL,PRMPT2
B91C CD55B9  CALL  DSPMSG
B91F CD925F  CALL  5F92H          ; screen editor
B922 23      INC  HL
B923 CD43B9  CALL  CLRFAC      ; clear floating point accumulator
B926 CDC826  CALL  26C8H      ; convert into binary code
B929 CDA021  CALL  21A0H      ; convert into integer
B92C 2A41EC  LD   HL,(0EC41H)      ; set 2nd argument into reg-hl
;
B92F D1      POP  DE          ; set 1st argument into reg-de
;
B930 CDAB23  CALL  23ABH          ;*division for integer
;
B933 CDD028  CALL  28D0H          ; convert into character code
B936 E5      PUSH HL
B937 217BB9  LD   HL,ANSWER
B93A CD55B9  CALL  DSPMSG
B93D E1      POP  HL
B93E CD55B9  CALL  DSPMSG          ; display result
;
B941 18BD    JR   START
;
; clear floating point accumulator subroutine
;
B943 E5      CLRFAC: PUSH HL
B944 210000  LD   HL,0
B947 223DEC  LD   (0EC3DH),HL
B94A 223FEC  LD   (0EC3FH),HL

```



```

B94D 2241EC      LD      (0EC41H),HL
B950 2243EC      LD      (0EC43H),HL
B953 E1          POP     HL
B954 C9          RET

;
; display message subroutine
;
B955 7E          DSPMSG:LD      A,(HL)
B956 A7          AND      A
B957 C8          RET      Z
B958 C0003E      CALL 3E00H      ; display a character
B95B 23          INC      HL
B95C 18F7        JR        DSPMSG

```

```

;
; message area
;
B95E 0D0A0A20    PRMPT1:DEFB 0DH,0AH,0AH,'      x%  ? ',0
B962 20202020
B966 78252020
B96A 203F2000
B96E 20202020    PRMPT2:DEFB '      y%  ? ',0
B972 20792520
B976 20203F20
B97A 00
B97B 7825202F    ANSWER:DEFB 'x% / y% --> ',0
B97F 20792520
B983 2D2D3E20
B987 00
;

```

a=&hb900:call a

```

      x%  ? 20
      y%  ? 6
x% / y% --> 3

      x%  ? 23.456
      y%  ? 10
x% / y% --> 2

      x%  ? -500
      y%  ? 20
x% / y% --> -25

      x%  ? 100
      y%  ? .1
Division by zero
Ok

```

## 2 3 F 7 H : 整数の符号を反転する.

アドレス	2 3 F 7 H
機能	FACCに格納された整数の符号を反転する.
レジスタ	A, F, B, C, D, E, H, L

**解 説** フローティング・アキュムレータ(FACC)に格納されている16ビット長の符号付き整数の符号を反転(NEGATION)し、レジスタHLおよびフローティング・アキュムレータ(FACC)に整数型として格納してもどります.

ただし、フローティング・アキュムレータ(FACC)に格納されていたデータが、8000H(-32768)であった場合には、符号反転(NEGATION)の結果が整数型をオーバーフローしてしまうため、演算結果(32768)はフローティング・アキュムレータ(FACC)に単精度の実数として格納されます.

具体的には、フローティング・アキュムレータ(EC41H~EC42H番地)のデータをレジスタHLにロード後、23EAH番地にエントリします.

### サンプル

```

;
; --- negation for integer ---
;
        ORG 0B900H
;
B900 2142B9   START: LD  HL,PROMPT
B903 C039B9   CALL  DSPMSG
B906 CD925F   CALL  5F92H           ; screen editor
B909 D8       RET  C             ; stop key, return to basic
B90A 23       INC  HL
B90B CD27B9   CALL  CLRFCAC       ; clear floating point accumulator
B90E CDC826   CALL  26C8H       ; convert into binary code
B911 CDA021   CALL  21A0H       ; convert into integer
;
B914 CDF723   CALL  23F7H           ; *negation for integer in facc
;
B917 CDD028   CALL  28D0H           ; convert into character code
B91A E5       PUSH HL
B91B 214EB9   LD   HL,ANSWER
B91E C039B9   CALL  DSPMSG
B921 E1       POP  HL
B922 C039B9   CALL  DSPMSG           ; display result
;
B925 18D9     JR   START
;
; clear floating point accumulator subroutine
;
B927 E5       CLRFCAC: PUSH HL
B928 210000   LD   HL,0
B92B 223DEC   LD   (0EC3DH),HL
B92E 223FEC   LD   (0EC3FH),HL
B931 2241EC   LD   (0EC41H),HL
B934 2243EC   LD   (0EC43H),HL
B937 E1       POP  HL
B938 C9       RET

```

```

;
; display message subroutine
;
B939 7E    DSPMSG:LD    A,(HL)
B93A A7          AND    A
B93B C8          RET    Z
B93C CD0D3E      CALL 3E0DH          ; display a character
B93F 23          INC    HL
B940 18F7        JR     DSPMSG
;
; message area
;
B942 0D0A0A20    PROMPT:DEFB 0DH,0AH,0AH,' x%  ? ',0
B946 78252020
B94A 203F2000
B94E 2D782520    ANSWER:DEFB '-x% --> ',0
B952 2D2D3E20
B956 00
;

```

a=&hb900:call a

```

x%  ? &h8000
-x% --> 32768

```

```

x%  ? 1234
-x% --> -1234

```

```

x%  ? -22222
-x% --> 22222

```

```

x%  ? 0.005
-x% --> 0

```

```

x%  ? 5.5
-x% --> -6

```

```

x%  ?
Ok

```

## 2 3 F A H : 整数の符号を反転する.

アドレス	2 3 F A H
機 能	整数の符号を反転する.
レジスタ	A, F, B, C, D, E, H, L

**解 説** レジスタHLに格納されている16ビット長の符号付き整数の符号を反転(NEGATION)し、レジスタHLおよびフローティング・アキュムレータ(FACC)に整数型として格納してもどります.

ただし、レジスタHLに格納されていたデータが、8000H(-32768)であった場合には、符号反転(NEGATION)の結果が整数型をオーバーフローしてしまうため、演算結果(32768)はフローティング・アキュムレータ(FACC)に単精度の実数として格納されます.

### サンプル

```

;
; --- negation for integer ---
;
        ORG 0B900H
;
B900 2145B9  START: LD  HL,PROMPT
B903 CD3CB9  CALL DSPMSG
B906 CD925F  CALL 5F92H          ; screen editor
B909 D8      RET  C          ; stop key, return to basic
B90A 23      INC HL
B90B CD2AB9  CALL CLRFBAC    ; clear floating point accumulator
B90E CDC826  CALL 26C8H      ; convert into binary code
B911 CDA021  CALL 21A0H      ; convert into integer
B914 2A41EC  LD HL,(0EC41H)   ; set an argument in register hl
;
B917 CDF A23  CALL 23FAH      ; *negation for integer in register
;
B91A CDD028  CALL 28D0H      ; convert into character code
B91D E5      PUSH HL
B91E 2151B9  LD HL,ANSWER
B921 CD3CB9  CALL DSPMSG
B924 E1      POP HL
B925 CD3CB9  CALL DSPMSG      ; display result
;
B928 18D6    JR  START
;
; clear floating point accumulator subroutine
;
B92A E5      CLRFBAC: PUSH HL
B92B 210000  LD HL,0
B92E 223DEC  LD (0EC3DH),HL
B931 223FEC  LD (0EC3FH),HL
B934 2241EC  LD (0EC41H),HL
B937 2243EC  LD (0EC43H),HL
B93A E1      POP HL
B93B C9      RET
;
; display message subroutine
;
B93C 7E      DSPMSG: LD A,(HL)
B93D A7      AND A
B93E C8      RET Z

```

```

B93F C00D3E      CALL 3E0DH      ; display a character
B942 23          INC HL
B943 18F7        JR DSPMSG
;
; message area
;
B945 0D0A0A20    PROMPT:DEFB 0DH,0AH,0AH,' x% ? ',0
B949 78252020
B94D 203F2000
B951 20782520    ANSWER:DEFB '-x% --> ',0
B955 202D3E20
B959 00
;

```

```

a=&hb900:call a

```

```

x% ? &h8000
-x% --> 32768

```

```

x% ? -22222
-x% --> 22222

```

```

x% ? 0.05
-x% --> 0

```

```

x% ? 0.5
-x% --> -1

```

```

x% ? 0.4
-x% --> 0

```

```

x% ? 33333
Overflow
Ok

```

## 240CH: 整数の除算を行ない剰余を求める.

アドレス	240CH
機能	整数の除算を行ない剰余を求める.
レジスタ	A, F, B, C, D, E, H, L

**解説** レジスタDEの符号付き16ビット整数を被除数とし、レジスタHLの符号付き16ビット整数を除数として除算を行ない、剰余をレジスタDEおよびフローティング・アキュムレータ(FACC)に格納してとりもどす.

ただし、レジスタHLに与える除数が、0000H(ゼロ)であった場合にはのみ、“Division by zero”(／0 Error)エラーが発生するためN<sub>88</sub>-BASICまたは機械語モニタの制御下に入ります.

### サンプル

```

;
; --- calculate remainder ---
;
;      ORG 0B900H
;
B900 215B89  START: LD  HL,PRMPT1
B903 CD55B9      CALL DSPMSG
B906 CD925F      CALL 5F92H          ; screen editor
B909 D8          RET C              ; stop key, return to basic
B90A 23          INC HL
B90B CD43B9      CALL CLRFBAC      ; clear floating point accumulator
B90E CDC826      CALL 26C8H        ; convert into binary code
B911 CDA021      CALL 21A0H        ; convert into integer
B914 ED5B41EC    LD DE,(0EC41H)    ; store 1st argument into reg-de
B918 D5          PUSH DE
;
B919 2170B9      LD HL,PRMPT2
B91C CD55B9      CALL DSPMSG
B91F CD925F      CALL 5F92H          ; screen editor
B922 23          INC HL
B923 CD43B9      CALL CLRFBAC      ; clear floating point accumulator
B926 CDC826      CALL 26C8H        ; convert into binary code
B929 CDA021      CALL 21A0H        ; convert into integer
B92C 2A41EC      LD HL,(0EC41H)    ; set 2nd argument into reg-hl
;
B92F D1          POP DE             ; set 1st argument into reg-de
;
B930 CD0C24      CALL 240CH         ; *calculate remainder
;
B933 CD0D28      CALL 28D0H         ; convert into character code
B936 E5          PUSH HL
B937 217FB9      LD HL,ANSWER
B93A CD55B9      CALL DSPMSG
B93D E1          POP HL
B93E CD55B9      CALL DSPMSG        ; display result
;
B941 18BD        JR START
;
; clear floating point accumulator subroutine
;
B943 E5          CLRFBAC: PUSH HL
B944 210000      LD HL,0
B947 2230EC      LD (0EC3DH),HL
B94A 223FEC      LD (0EC3FH),HL

```

```

B94D 2241EC      LD      (0EC41H),HL
B950 2243EC      LD      (0EC43H),HL
B953 E1          POP     HL
B954 C9          RET

;
; display message subroutine
;
B955 7E          DSPMSG:LD      A,(HL)
B956 A7          AND      A
B957 C8          RET      Z
B958 CD0D3E      CALL 3E0DH      ; display a character
B95B 23          INC      HL
B95C 18F7        JR      DSPMSG

;
; message area
;
B95E 0D0A0A20    PRMPT1:DEFB 0DH,0AH,0AH,'      x%  ? ',0
B962 20202020
B966 20207825
B96A 2020203F
B96E 2000
B970 20202020    PRMPT2:DEFB      y%  ? ',0
B974 20202079
B978 25202020
B97C 3F2000
B97F 72656D61    ANSWER:DEFB 'remainder --> ',0
B983 696E6465
B987 72202D2D
B98B 3E2000
;

```

```

a=&hb900:call a

```

```

      x%  ? 10
      y%  ? 3
remainder --> 1

```

```

      x%  ? 1000
      y%  ? 501
remainder --> 499

```

```

      x%  ? 10
      y%  ? -3
remainder --> 1

```

```

      x%  ? 1000
      y%  ? -501
remainder --> 499

```

```

      x%  ?
Ok

```



## 2 4 1 D H : 倍精度型実数の減算を行なう。

アドレス	2 4 1 D H
機 能	倍精度型実数の減算を行なう。
レジスタ	A, F, B, C, D, E, H, L

**解 説** フローティング・アキュムレータ( E C 3 D H ~ E C 4 4 H 番地) に格納された倍精度型の実数を被減数とし、サブ・フローティング・アキュムレータ( E C 4 A H ~ E C 5 1 H 番地) に格納された倍精度型の実数を減数として減算を行ない、フローティング・アキュムレータに差を格納してもどります。

### サンプル

```

;
; --- subtraction for double precision ---
;
;      ORG 0B900H
;
B900 2177B9  START: LD HL,PRMPT1
B903 CD6EB9      CALL DSPMSG
B906 CD925F      CALL 5F92H          ; screen editor
B909 D8          RET C          ; stop key, return to basic
B90A 23          INC HL
B90B CD5CB9      CALL CLRFBAC      ; clear floating point accumulator
B90E CDC826      CALL 26C8H      ; convert into binary code
B911 CD3E22      CALL 223EH      ; convert into double precision
;
B914 213DEC      LD HL,0EC3DH      ;
B917 11A1B9      LD DE,BUFFER      ; store 1st argument into buffer
B91A 010800      LD BC,8          ;
B91D EDB0        LDIR
;
B91F 2187B9      LD HL,PRMPT2
B922 CD6EB9      CALL DSPMSG
B925 CD925F      CALL 5F92H          ; screen editor
B928 D8          RET C          ; stop key, return to basic
B929 23          INC HL
B92A CD5CB9      CALL CLRFBAC      ; clear floating point accumulator
B92D CDC826      CALL 26C8H      ; convert into binary code
B930 CD3E22      CALL 223EH      ; convert into double precision
;
B933 213DEC      LD HL,0EC3DH      ;
B936 114AEC      LD DE,0EC4AH      ; set 2nd argument into sub-facc
B939 010800      LD BC,8          ;
B93C EDB0        LDIR
;
B93E 21A1B9      LD HL,BUFFER      ;
B941 113DEC      LD DE,0EC3DH      ; set 1st argument into facc
B944 010800      LD BC,8          ;
B947 EDB0        LDIR
;
B949 CD1024      CALL 241DH          ;*subtraction for double precision
;
B94C CD0028      CALL 28D0H          ; convert into character code
B94F E5          PUSH HL
B950 2194B9      LD HL,ANSWER
B953 CD6EB9      CALL DSPMSG
B956 E1          POP HL
B957 CD6EB9      CALL DSPMSG          ; display result
;
B95A 18A4        JR START

```



## 2 4 2 4 H : 倍精度型実数の加算を行なう.

アドレス	2 4 2 4 H
機能	倍精度型実数の加算を行なう.
レジスタ	A, F, B, C, D, E, H, L

**解 説** フローティング・アキュムレータ (EC 3 DH ~ EC 4 4 H 番地) に格納された倍精度型の実数を被加数とし, サブ・フローティング・アキュムレータ (EC 4 AH ~ EC 5 1 H 番地) に格納された倍精度型の実数を加数として加算を行ない, フローティング・アキュムレータに和を格納してもどります.

### サンプル

```

; --- addition for double precision ---
;
; ORG 0B900H
;
B900 2177B9 START: LD HL,PRMPT1
B903 CD6EB9 CALL DSPMSG
B906 CD925F CALL 5F92H ; screen editor
B909 D8 RET C ; stop key, return to basic
B90A 23 INC HL
B90B CD5CB9 CALL CLRFAQ ; clear floating point accumulator
B90E CDC826 CALL 26C8H ; convert into binary code
B911 CD3E22 CALL 223EH ; convert into double precision
;
B914 213DEC LD HL,0EC3DH
B917 11A1B9 LD DE,BUFFER
B91A 010800 LD BC,8
B91D EDB0 LDIR
;
B91F 2187B9 LD HL,PRMPT2
B922 CD6EB9 CALL DSPMSG
B925 CD925F CALL 5F92H ; screen editor
B928 D8 RET C ; stop key, return to basic
B929 23 INC HL
B92A CD5CB9 CALL CLRFAQ ; clear floating point accumulator
B92D CDC826 CALL 26C8H ; convert into binary code
B930 CD3E22 CALL 223EH ; convert into double precision
;
B933 213DEC LD HL,0EC3DH
B936 114AEC LD DE,0EC4AH
B939 010800 LD BC,8
B93C EDB0 LDIR
;
B93E 21A1B9 LD HL,BUFFER
B941 113DEC LD DE,0EC3DH
B944 010800 LD BC,8
B947 EDB0 LDIR
;
B949 CD2424 CALL 2424H ;*addition
;
B94C CDD028 CALL 28D0H ; convert into character code
B94F E5 PUSH HL
B950 2194B9 LD HL,ANSWER
B953 CD6EB9 CALL DSPMSG
B956 E1 POP HL
B957 CD6EB9 CALL DSPMSG ; display result
;
B95A 18A4 JR START

```

```

;
; clear floating point accumulator subroutine
;
B95C E5      CLRFAC: PUSH HL
B95D 210000      LD HL,0
B960 223DEC      LD (0EC3DH),HL
B963 223FEC      LD (0EC3FH),HL
B966 2241EC      LD (0EC41H),HL
B969 2243EC      LD (0EC43H),HL
B96C E1          POP HL
B96D C9          RET

;
; display message subroutine
;
B96E 7E      DSPMSG: LD A,(HL)
B96F A7      AND A
B970 C8      RET Z
B971 CD0D3E      CALL 3E0DH ; display a character
B974 23          INC HL
B975 18F7      JR DSPMSG

;
; message area
;
B977 0D0A0A20 PRMPT1: DEFB 0DH,0AH,0AH,' x# ? ',0
B97B 20202020
B97F 78232020
B983 203F2000
B987 20202020 PRMPT2: DEFB ' y# ? ',0
B98B 20792320
B98F 20203F20
B993 00
B994 7823202B ANSWER: DEFB 'x# + y# --> ',0
B998 20792320
B99C 2D2D3E20
B9A0 00

;
; buffer for the first argument
;
B9A1      BUFFER: DEFS 8
;

```

a=&hb900:call a

```

x# ? 12345678
y# ? 1234567800000000
x# + y# --> 1234567812345678

x# ? 1
y# ? 2
x# + y# --> 3

x# ? 0.0000000000001
y# ? 0.0000000000001
x# + y# --> .0000000000002

x# ? 0.0000000000000000000000000000000000000000001
y# ? 0
x# + y# --> 1D-37

x# ?
Ok

```

## 2 5 5 3 H：倍精度型実数の乗算を行なう。

アドレス	2 5 5 3 H
機能	倍精度型実数の乗算を行なう。
レジスタ	A, F, B, C, D, E, H, L

**解 説** フローティング・アキュムレータ(EC3DH～EC44H番地)に格納された倍精度型の実数を被乗数とし、サブ・フローティング・アキュムレータ(EC4AH～EC51H番地)に格納された倍精度型の実数を乗数として乗算を行ない、フローティング・アキュムレータに積を格納してもどります。

### サンプル

```

;
; --- multiplication for double precision ---
;
      ORG 0B900H
;
START: LD  HL,PRMPT1
      CALL DSPMSG
      CALL 5F92H
      RET  C
      INC  HL
      CALL CLRFAC
      CALL 26C8H
      CALL 223EH
;
      LD  HL,0EC3DH
      LD  DE,BUFFER
      LD  BC,8
      LDIR
;
      LD  HL,PRMPT2
      CALL DSPMSG
      CALL 5F92H
      RET  C
      INC  HL
      CALL CLRFAC
      CALL 26C8H
      CALL 223EH
;
      LD  HL,0EC3DH
      LD  DE,0EC4AH
      LD  BC,8
      LDIR
;
      LD  HL,BUFFER
      LD  DE,0EC3DH
      LD  BC,8
      LDIR
;
      CALL 2553H
;
      CALL 28D0H
      PUSH HL
      LD  HL,ANSWER
      CALL DSPMSG
      POP  HL
      CALL DSPMSG
;
      JR  START

```

```

;
; clear floating point accumulator subroutine
;
B95C E5      CLRAC:PUSH HL
B95D 210000      LD HL,0
B960 223DEC      LD (0EC3DH),HL
B963 223FEC      LD (0EC3FH),HL
B966 2241EC      LD (0EC41H),HL
B969 2243EC      LD (0EC43H),HL
B96C E1          POP HL
B96D C9          RET

;
; display message subroutine
;
B96E 7E          DSPMSG:LD A,(HL)
B96F A7          AND A
B970 C8          RET Z
B971 CD0D3E      CALL 3E0DH ; display a character
B974 23          INC HL
B975 18F7        JR DSPMSG

;
; message area
;
B977 0D0A0A20    PRMPT1:DEFB 0DH,0AH,0AH,' x# ? ',0
B978 20202020
B97F 78232020
B983 203F2000
B987 20202020    PRMPT2:DEFB ' y# ? ',0
B988 20792320
B98F 20203F20
B993 00
B994 7823202A    ANSWER:DEFB 'x# * y# --> ',0
B998 20792320
B99C 2D2D3E20
B9A0 00

;
; buffer for the first argument
;
B9A1          BUFFER:DEFS 8
;

```

a=&hb900:call a

```

x# ? 10000000
y# ? 10000000
x# * y# --> 1000000000000000

x# ? 0.0001234567890
y# ? 10000000000000
x# * y# --> 123456789

x# ? 3
y# ? -0.5
x# * y# --> -1.5

x# ? 0
y# ? 0
x# * y# --> 0

x# ?

```

Ok



## 2 6 2 9 H : 倍精度型実数の除算を行なう.

アドレス	2 6 2 9 H
機 能	倍精度型実数の除算を行なう.
レジスタ	A, F, B, C, D, E, H, L

**解 説** フローティング・アキュムレータ (EC 3 DH ~ EC 4 H 番地) に格納された倍精度型の実数を被除数とし, サブ・フローティング・アキュムレータ (EC 4 AH ~ EC 5 1 H 番地) に格納された倍精度型の実数を除数として除算を行ない, フローティング・アキュムレータに商を格納してもどります.

もし, フローティング・アキュムレータに与える除数が 0 (ゼロ) であった場合には, "Division by zero" ( / 0 Error) エラーとなりますがインタプリタは, " / 0 " のメッセージを表示し, 単精度型の実数として扱うことのできる最大の数値 ( 1 . 7 0 1 4 1 E + 3 8 ) を除算の結果として与えます.

### サンプル

```

;
; --- division for double precision ---
;
      ORG 0B900H
;
B900 2177B9 ; START: LD HL,PRMPT1
B903 CD6EB9 CALL DSPMSG
B906 CD925F CALL 5F92H ; screen editor
B909 08 RET C ; stop key, return to basic
B90A 23 INC HL
B90B CD5CB9 CALL CLRFAF ; clear floating point accumulator
B90E CDC826 CALL 26C8H ; convert into binary code
B911 CD3E22 CALL 223EH ; convert into double precision
;
B914 213DEC LD HL,0EC3DH ;
B917 11A1B9 LD DE,BUFFER ;
B91A 010800 LD BC,8 ; store 1st argument into buffer
B91D EDB0 LDIR ;
;
B91F 2187B9 LD HL,PRMPT2
B922 CD6EB9 CALL DSPMSG
B925 CD925F CALL 5F92H ; screen editor
B928 08 RET C ; stop key, return to basic
B929 23 INC HL
B92A CD5CB9 CALL CLRFAF ; clear floating point accumulator
B92D CDC826 CALL 26C8H ; convert into binary code
B930 CD3E22 CALL 223EH ; convert into double precision
;
B933 213DEC LD HL,0EC3DH ;
B936 114AEC LD DE,0EC4AH ;
B939 010800 LD BC,8 ; set 2nd argument into sub-facc
B93C EDB0 LDIR ;
;
B93E 21A1B9 LD HL,BUFFER ;
B941 113DEC LD DE,0EC3DH ;
B944 010800 LD BC,8 ; set 1st argument into facc
B947 EDB0 LDIR ;
;
B949 CD2926 CALL 2629H ; *division for double precision

```



```

;
B94C CDD028      CALL 28D0H      ; convert into character code
B94F E5          PUSH HL
B950 2194B9      LD HL,ANSWER
B953 CD6EB9      CALL DSPMSG
B956 E1          POP HL
B957 CD6EB9      CALL DSPMSG      ; display result
;
B95A 18A4        JR START
;
; clear floating point accumulator subroutine
;
B95C E5          CLRFPAC:PUSH HL
B95D 210000      LD HL,0
B960 223DEC      LD (0EC3DH),HL
B963 223FEC      LD (0EC3FH),HL
B966 2241EC      LD (0EC41H),HL
B969 2243EC      LD (0EC43H),HL
B96C E1          POP HL
B96D C9          RET
;
; display message subroutine
;
B96E 7E          DSPMSG:LD A,(HL)
B96F A7          AND A
B970 C8          RET Z
B971 CD0D3E      CALL 3E0DH      ; display a character
B974 23          INC HL
B975 18F7        JR DSPMSG
;
; message area
;
B977 0D0A0A20    PRMPT1:DEFB 0DH,0AH,0AH,' x# ? ',0
B97B 20202020
B97F 78232020
B983 203F2000
B987 20202020    PRMPT2:DEFB ' y# ? ',0
B98B 20792320
B98F 20203F20
B993 00
B994 7823202F    ANSWER:DEFB 'x# / y# --> ',0
B998 20792320
B99C 2D2D3E20
B9A0 00
;
; buffer for the first argument
;
B9A1             BUFFER:DEFS 8
;

```

a=&hb900:call a

```

x# ? 20
y# ? 6
x# / y# --> 3.3333333333333333

x# ? 20
y# ? 3
x# / y# --> 6.6666666666666667

x# ? 1.23456789012345
y# ? 1000
x# / y# --> 1.23456789012345D-03

x# ? 300
y# ? 0
/0
x# / y# --> 1.701411834604693D+38

x# ?
Ok

```

## 2 6 B 5 H : 文字列を倍精度型実数に変換する.

アドレス	2 6 B 5 H
機 能	文字列を倍精度型実数に変換する.
レジスタ	A, F, B, C, D, E, H, L

**解 説** レジスタ H L で指定するアドレスから格納されている, キャラクタ・コードによる数値データを, 倍精度実数型のバイナリ, コードに変換して, フローティング・アキュムレータに与えてもどります. 通常のデータであれば全て区別なく倍精度型実数として格納され, E A B D H 番地にも倍精度実数型を表す 0 8 H が入りますが, 次の条件を満たすデータのみは整数型または単精度実数型の数値として変換されます.

- 1) 頭に & O または & を伴った 8 進形式のデータ
- 2) 頭に & H を伴った 1 6 進形式のデータ
- 3) 末尾に % を伴った整数型のデータ
- 4) 末尾に ! を伴った単精度実数型のデータ
- 5) E を使った指数形式のデータ

上記の条件が満たされていなければ, たとえば 1 0 であっても整数型とは見なされません. 1 0 % であれば整数型となります.

また, データの中に数値に関係のないキャラクタ・コード ( 0 0 H を含む ) が存在すれば, 直前のキャラクタまでが有効となります.

### サンプル

```

;
; --- input and display number ---
;
; ORG 0B900H
;
B900 213AB9 START: LD HL,PROMPT
B903 CD31B9 CALL DSPMSG
B906 CD925F CALL 5F92H ; screen editor
B909 D8 RET C ; stop key, return to basic
B90A 23 INC HL
B90B CDB526 CALL 26B5H ; *double precision binary code
;
B90E F7 RST 30H ; check type of facc
;
B90F F217B9 JP P,NOT3
B912 2149B9 LD HL,TYPE2 ; integer
B915 180F JR DISPLAY
;
B917 EA1FB9 NOT3: JP PE,NOT4
B91A 2155B9 LD HL,TYPE4 ; single precision number
B91D 1807 JR DISPLAY
;
B91F 38DF NOT4: JR C,START ; error, input a number again
B921 2171B9 LD HL,TYPE8 ; double precision number
B924 1800 JR DISPLAY

```

```

B926 C031B9      ; DSPLAY:CALL DSPMSG
;
B929 C0D028      ; CALL 2800H          ; convert into character code
B92C C031B9      CALL DSPMSG
;
B92F 18CF        JR    START
;
; display message subroutine
;
B931 7E          DSPMSG:LD    A,(HL)
B932 A7          AND    A
B933 C8          RET    Z
B934 C0D03E      CALL 3E00H          ; display a character
B937 23          INC    HL
B938 18F7        JR    DSPMSG
;
; message area
;
B93A 000A0A61    PROMPT:DEFB 0DH,0AH,0AH,'a number : ',0
B93E 206E756D
B942 62657220
B946 3A2000
B949 696E7465    TYPE2: DEFB 'integer of ',0
B94D 67657220
B951 6F662000
B955 73696E67    TYPE4: DEFB 'single precision number of ',0
B959 6C652070
B95D 72656369
B961 73696F6E
B965 206E756D
B969 62657220
B96D 6F662000
B971 646F7562    TYPE8: DEFB 'double precision number of ',0
B975 6C652070
B979 72656369
B97D 73696F6E
B981 206E756D
B985 62657220
B989 6F662000
;

```

## 2 6 B C H : 文字列を数値データに変換する.

アドレス	2 6 B C H
機 能	文字列を数値データに変換する.
レジスタ	A, F, B, C, D, E, H, L

**解 説** レジスタHLで指定するアドレスから格納されている, キャラクタ・コードによる数値データを, 倍精度実数型, 単精度実数型または整数型のバイナリ・コードに変換して, フローティング・アキュムレータ (F A C C) に格納し, E A B D H番地には, 型を示す数値 (0 8 H, 0 4 Hまたは0 2 H) が入ります. データの中に数値に関係のないキャラクタ・コード (0 0 Hを含む) が存在する場合には, 直前のキャラクタまでが有効となります.

なお, 本書掲載のサンプル・プログラムの一部は, 2 6 B C H番地のかわりに 2 6 C 8 H番地をコールしていますが, 2 6 C 8 H番地のサブルーチンではフローティング・アキュムレータの型を示すE A B D H番地が正常に代入されません. このため, 事前にフローティング・アキュムレータ (F A C C) を0 0 Hで埋めておく等の配慮が必要です.

### サンプル

```

;
; --- input and display number ---
;
;       ORG 0B900H
;
B900 213AB9 ; START: LD HL,PROMPT
B903 CD31B9 ;       CALL DSPMSG
B906 CD925F ;       CALL 5F92H ; screen editor
B909 D8 ;       RET C ; stop key, return to basic
B90A 23 ;       INC HL
B90B CDBC26 ;       CALL 26BCH ; *convert into binary code
;
B90E F7 ;       RST 30H ; check type of facc
;
B90F F217B9 ;       JP P,NOT3
B912 2149B9 ;       LD HL,TYPE2 ; integer
B915 180F ;       JR DISPLAY
;
B917 EA1FB9 ; NOT3: JP PE,NOT4
B91A 2155B9 ;       LD HL,TYPE4 ; single precision number
B91D 1807 ;       JR DISPLAY
;
B91F 38DF ; NOT4: JR C,START ; error, input a number again
B921 2171B9 ;       LD HL,TYPE8 ; double precision number
B924 1800 ;       JR DISPLAY
;
B926 CD31B9 ; DISPLAY:CALL DSPMSG
;
B929 CD0028 ;       CALL 28D0H ; convert into character code
B92C CD31B9 ;       CALL DSPMSG
;
B92F 18CF ;       JR START
;
; display message subroutine

```



## 28C2H: 符号無し16ビットの10進数を入力する。

アドレス	28C2H
機能	符号無し16ビットの10進数を入力する。
レジスタ	A, F, B, C, D, E, H, L

**解説** レジスタHLのデータを符号無し16ビットの10進数として、CRTスクリーン上のカーソル位置に表示してもどります。

N88-BASIC では行番号の表示等に、このサブルーチンを利用しています。

なお、E64CH番地の出力フラグを01Hにセットすれば、プリンタへ出力します。

### サンプル

```

;                                     %12  b)
; --- display numbers with two bytes ---
;
;      ORG 0B900H
B900 210000      ;      LD  HL,0
;
;      LOOP: CALL DSPHL          ; display hexa decimal
;
B906 3E48      LD  A,'H'
B908 CD0D3E    CALL 3E0DH
B908 3E20      LD  A,' '
B90D CD0D3E    CALL 3E0DH
B910 3E3D      LD  A,'='
B912 CD0D3E    CALL 3E0DH
B915 3E20      LD  A,' '
B917 CD0D3E    CALL 3E0DH
;
B91A E5        PUSH HL
B91B CDC228    CALL 28C2H      ;*display decimal
B91E E1        POP  HL
;
B91F 3E0D      LD  A,0DH
B921 CD0D3E    CALL 3E0DH
B924 3E0A      LD  A,0AH
B926 CD0D3E    CALL 3E0DH
;
B929 CD36B9    CALL WAIT      ; wait a moment
;
B92C 23        INC  HL
B92D 7C        LD  A,H
B92E B5        OR   L
B92F C8        RET  Z
;
B930 CDC235    CALL 35C2H      ; check stop key
B933 30CE      JR   NC,LOOP    ; not stop key, jump to loop
;
B935 C9        RET            ; return to basic
;
; wait subroutine
;
B936 01FFFF    WAIT: LD  BC,0FFFFH
B939 08        WAIT1: DEC  BC
B93A 78        LD  A,B
B93B B1        OR   C
B93C 20FB      JR   NZ,WAIT1
B93E C9        RET

```

```

;
; display hexa decimal number in register hl subroutine
;
; destroys : register a,f,b
;
B93F 7C      DSPHL: LD    A,H
B940 CD44B9  CALL DSPACC
B943 7D      LD    A,L
;
B944 47      DSPACC:LD    B,A
B945 0F      RRCA
B946 0F      RRCA
B947 0F      RRCA
B948 0F      RRCA
B949 CD4DB9  CALL HALF
B94C 78      LD    A,B
;
B94D E60F    HALF: AND    0FH
B94F FE0A    CP    10
B951 3802    JR    C,NUMBER
B953 C607    ADD    A,7
;
B955 C630    NUMBER:ADD    A,30H
B957 CD0D3E  CALL 3E0DH
B95A C9      RET
;
; display a character

```

a=&hb900:call a

```

0000H = 0
0001H = 1
0002H = 2
0003H = 3
0004H = 4
0005H = 5
0006H = 6
0007H = 7
0008H = 8
0009H = 9
000AH = 10
000BH = 11
000CH = 12
000DH = 13
000EH = 14
000FH = 15
0010H = 16
0011H = 17
0012H = 18
^C

```

Break  
Ok



## 2 8 D 0 H : 数値データを文字列に変換する.

アドレス	2 8 D 0 H
機 能	数値データを文字列に変換する.
レジスタ	A, F, B, C, D, E, H, L

**解 説** E A B D H 番地の型を示す数字 ( 2 , 4 または 8 ) により, フローティング・アキュムレータ ( F A C C ) に格納されているバイナリ形式の数値データをキャラクタ・コードの文字列に変換して E C 5 2 H 番地から 3 5 バイトのバッファに格納し, データの末尾に 0 0 H を付け加えます.

リターン時には, 文字列の先頭アドレスがレジスタ H L に入っています.

具体的にはアキュムレータに 0 0 H を入れて (フリー・フォーマットを指定) 2 8 D 1 H 番地にエントリします.

## サンプル

```

;
; --- input and display a number free format ---
;
      ORG 0B900H
;
B900 2128B9 START: LD HL,PROMPT
B903 CD1FB9 CALL DSPMSG
B906 CD925F CALL 5F92H ; screen editor
B909 D8 RET C ; stop key, return to basic
B90A 23 INC HL
B90B CDBC26 CALL 26BCH ; convert into binary code
;
B90E CDD028 CALL 28D0H ;*convert facc into character code
B911 E5 PUSH HL
B912 2139B9 LD HL,ANSWER
B915 CD1FB9 CALL DSPMSG
B918 E1 POP HL
B919 CD1FB9 CALL DSPMSG ; display the number
;
B91C C300B9 JP START
;
; display message subroutine
;
; destroys : register a,f,h,l
;
B91F 7E DSPMSG:LD A,(HL)
B920 A7 AND A
B921 C8 RET Z
B922 CD0D3E CALL 3E0DH ; display a character
B925 23 INC HL
B926 18F7 JR DSPMSG
;
; message area
;
B928 0D0A0A74 PROMPT:DEFB 0DH,0AH,0AH,'the number ? ',0
B92C 6865206E
B930 756D6265
B934 72203F20
B938 00
B939 74686520 ANSWER:DEFB 'the number : ',0
B93D 6E756D62
B941 6572203A
B945 2000
;

```

a=&hb900:call a

the number ? 5  
the number : 5

the number ? 5.5  
the number : 5.5

the number ? 5#  
the number : 5

the number ? &ha  
the number : 10

the number ? &o777  
the number : 511

the number ? 1.12345  
the number : 1.12345

the number ?  
Ok

## 28D1H：フォーマットを付けて文字列に変換する。

アドレス

28D1H

機能

数値データをフォーマットを付けた文字列に変換する。

レジスタ

A, F, B, C, D, E, H, L

解説

EABDH番地の型を示す数字(2, 4 または 8)により, フローティング・アキュムレータ(FACC)に格納されているバイナリ形式の数値データをキャラクタ・コードの文字列に変換してEC52H番地から35バイトのバッファに格納し, データの末尾に00Hを付け加えます。

リターン時には, 文字列の先頭アドレスがレジスタHLに入っています。

このサブルーチンでは, 文字列データの生成時に様々なフォーマットを指定する事ができるため, BASICのPRINT USING, LPRINT USING, FORTRANのFORMATと同様の機能を比較的簡単に実現する事ができます。

フォーマットの指定は, アキュムレータおよびレジスタB, レジスタCにパラメータを与えて行ないますが, 以下にそれぞれのパラメータの意味を示します。

### 1) アキュムレータ

ビット7——フォーマット指定を行なうか否かを指定します。すなわち, このビットが0の場合には以下のフォーマット指定は全て無意味なものとなります。

ビット6——このビットを1にセットした場合, 数値の整数部が3桁毎にカンマ(,)で区切られます。

ビット5——このビットを1にセットした場合, 数値領域の左側に空白部分ができたとき, そこをアスタリスク(\*)で埋めます。

ビット4——このビットを1にセットした場合, 数値の直前にエンマーク(¥)を付け加えます。

ビット3——このビットを1にセットした場合, 数値が負の時のマイナス符号(-)だけでなく, 数値が正の時のプラス符号(+)も, 数値の前に付加します。

ビット2——このビットを1にセットした場合, マイナス符号(-)またはプラス符号(+)を数値の直後に付加します。

ビット0——このビットを1にセットした場合, 数値は指数形式で変換されます。

## 2) レジスタ B

小数点より左（整数部分）の桁数を指定します。指定した桁数より数値の桁数が小さい場合には、右づめになります。

## 3) レジスタ C

小数点のフィールド 1 桁分を含む、小数点より右（小数部分）の桁数を指定します。

指定した領域より数値の桁数が大きい場合、数値の直前にパーセント・マーク（%）が付加されます。数値の丸めが領域より大きくなる原因となった場合も、丸めた数値の前にパーセント・マーク（%）が付加されます。

### サンプル

```

; --- print using test program ---
;
;      ORG 0B900H
;
; input the number section
;
B900 21DD89  START: LD  HL,PRMPT1
B903 CDD4B9      CALL DSPMSG
B906 CD925F      CALL 5F92H          ; screen editor
B909 08          RET C              ; stop key, return to basic
B90A 23          INC HL
B90B CDC2B9      CALL CLRAC          ; clear floating point accumulator
B90E CDC826      CALL 26C8H          ; convert into binary code
;
; print using section
;
B911 AF          XOR  A              ; clear accumulator
B912 5F          LD   E,A            ; clear register e
;
B913 21F8B9      LD   HL,PRMPT7      ; using ?
B916 CDD4B9      CALL DSPMSG
B919 CD8335      CALL 3583H          ; input a character from keyboard
B91C CD0D3E      CALL 3E0DH          ; echo-back a character
B91F FE79        CP   'y'
B921 C2B0B9      JP   NZ,DISPLAY     ; display in free format
B924 CBF8        SET  7,E            ; display in not free format
;
B926 2111BA      COMMA: LD  HL,PRMPT6 ; commas ?
B929 CDD4B9      CALL DSPMSG
B92C CD8335      CALL 3583H          ; input a character from keyboard
B92F CD0D3E      CALL 3E0DH          ; echo-back a character
B932 FE79        CP   'y'
B934 2002        JR   NZ,ASTAR       ; not display commas
B936 CBF3        SET  6,E            ; display commas
;
B938 2128BA      ASTAR: LD  HL,PRMPT5 ; asterisk ?
B93B CDD4B9      CALL DSPMSG
B93E CD8335      CALL 3583H          ; input a character from keyboard
B941 CD0D3E      CALL 3E0DH          ; echo-back a character
B944 FE79        CP   'y'
B946 2002        JR   NZ,YEN         ; not display asterisks
B948 CBE8        SET  5,E            ; display asterisks
;
B94A 2145BA      YEN:  LD  HL,PRMPT4 ; yen mark ?
B94D CDD4B9      CALL DSPMSG
B950 CD8335      CALL 3583H          ; input a character from keyboard
B953 CD0D3E      CALL 3E0DH          ; echo-back a character
B956 FE79        CP   'y'

```

```

B958 2002      JR  NZ,PLUS      ; not display yen mark
B95A CBE3      SET  4,E         ; display yen mark
;
B95C 215FBA    PLUS: LD  HL,PRMPT3 ; plus mark ?
B95F CDD4B9    CALL DSPMSG
B962 CD8335    CALL 3583H
B965 CD0D3E    CALL 3E0DH        ; input a character from keyboard
B968 FE79      CP   'y'         ; echo-back a character
B96A 2002      JR  NZ,SIGN      ; not display plus mark
B96C CBD8      SET  3,E         ; display plus mark
;
B96E 2179BA    SIGN: LD  HL,PRMPT2 ; sign ?
B971 CDD4B9    CALL DSPMSG
B974 CD8335    CALL 3583H        ; input a character from keyboard
B977 CD0D3E    CALL 3E0DH        ; echo-back a character
B97A FE79      CP   'y'
B97C 2002      JR  NZ,INDEX     ; not display sign after a number
B97E CBD3      SET  2,E         ; display sign after a number
;
B980 2193BA    INDEX: LD  HL,PRMPT0 ; index format ?
B983 CDD4B9    CALL DSPMSG
B986 CD8335    CALL 3583H        ; input a character from keyboard
B989 CD0D3E    CALL 3E0DH        ; echo-back a character
B98C FE79      CP   'y'
B98E 2002      JR  NZ,LEFT      ; not in index format
B990 CBC3      SET  0,E         ; in index format
;
B992 21ADBA    LEFT: LD  HL,PRMPTL ; left of point ?
B995 CDD4B9    CALL DSPMSG
B998 CD8335    CALL 3583H        ; input a character from keyboard
B99B CD0D3E    CALL 3E0DH        ; echo-back a character
B99E D630      SUB  '0'         ; convert a character into binary
B9A0 47        LD  B,A
;
B9A1 21C7BA    RIGHT: LD  HL,PRMPTR ; right of point ?
B9A4 CDD4B9    CALL DSPMSG
B9A7 CD8335    CALL 3583H        ; input a character from keyboard
B9AA CD0D3E    CALL 3E0DH        ; echo-back a character
B9AD D630      SUB  '0'         ; convert a character into binary
B9AF 4F        LD  C,A
;
B9B0 7B        DISPLAY:LD  A,E    ; set the most important agument
B9B1 CDD128    CALL 28D1H        ;*convert facc into character code
B9B4 E5        PUSH HL
B9B5 21E1BA    LD  HL,ANSWER
B9B8 CDD4B9    CALL DSPMSG
B9BB E1        POP  HL
B9BC CDD4B9    CALL DSPMSG      ; display the number
;
B9BF C300B9    JP  START
;
; clear floating point accumulator subroutine
;
; destroys : none
;
B9C2 E5        CLRFAC:PUSH HL
B9C3 210000    LD  HL,0
B9C6 223DEC    LD  (0EC3DH),HL
B9C9 223FEC    LD  (0EC3FH),HL
B9CC 2241EC    LD  (0EC41H),HL
B9CF 2243EC    LD  (0EC43H),HL
B9D2 E1        POP  HL
B9D3 C9        RET
;
; display message subroutine
;
; destroys : register a,f,h,l
;
B9D4 7E        DSPMSG:LD  A,(HL)
B9D5 A7        AND  A
B9D6 C8        RET  Z
B9D7 CD0D3E    CALL 3E0DH        ; display a character
B9DA 23        INC  HL
B9DB 18F7      JR  DSPMSG

```

```

;
; message area
;
B9D0 0D0A0A74 PRMPT1:DEFB 0DH,0AH,0AH,'the number ? ',0
B9E1 6865206E
B9E5 756D6265
B9E9 72203F20
B9ED 20202020
B9F1 20202020
B9F5 202000
;
B9F8 0D757369 PRMPT7:DEFB 0DH, 'using (y) ? ',0
B9FC 6E672028
BA00 7929203F
BA04 20202020
BA08 20202020
BA0C 20202020
BA10 00
BA11 0D0A636F PRMPT6:DEFB 0DH,0AH, 'commas (y) ? ',0
BA15 6D6D6173
BA19 20287929
BA1D 203F2020
BA21 20202020
BA25 20202020
BA29 2000
BA2B 0D0A6173 PRMPT5:DEFB 0DH,0AH, 'astarisks (y) ? ',0
BA2F 74617269
BA33 73687320
BA37 28792920
BA3B 3F202020
BA3F 20202020
BA43 2000
BA45 0D0A7965 PRMPT4:DEFB 0DH,0AH, 'yen mark (y) ? ',0
BA49 6E206D61
BA4D 726B2028
BA51 7929203F
BA55 20202020
BA59 20202020
BA5D 2000
BA5F 0D0A706C PRMPT3:DEFB 0DH,0AH, 'plus mark (y) ? ',0
BA63 7573206D
BA67 61726B20
BA6B 28792920
BA6F 3F202020
BA73 20202020
BA77 2000
BA79 0D0A7369 PRMPT2:DEFB 0DH,0AH, 'sign (y) ? ',0
BA7D 676E2028
BA81 7929203F
BA85 20202020
BA89 20202020
BA8D 20202020
BA91 2000
BA93 0D0A696E PRMPT0:DEFB 0DH,0AH, 'index format (y) ? ',0
BA97 64657820
BA9B 666F726D
BA9F 61742028
BAA3 7929203F
BAA7 20202020
BAAB 2000
;
BAAD 0D0A6C65 PRMPTL:DEFB 0DH,0AH, 'left of point (0-9) ? ',0
BAB1 6674206F
BAB5 6620706F
BAB9 696E7420
BABD 28302D39
BAC1 29203F20
BAC5 2000
BAC7 0D0A7269 PRMPTR:DEFB 0DH,0AH, 'right of point (0-9) ? ',0
BACB 67687420
BACF 6F662070
BAD3 6F696E74
BAD7 2028302D
BADB 3929203F

```

BADF 2000

```

;
BAE1 0D0A7468 ANSWER:DEFB 0DH,0AH, 'the number : ',0
BAE5 65206E75
BAE9 6D626572
BAED 203A2020
BAF1 20202020
BAF5 20202020
BAF9 2000
;
```

a=&hb900:call a

```

the number ?          1234.56789
using (y) ?           y
commas (y) ?           y
astarisks (y) ?        y
yen mark (y) ?         y
plus mark (y) ?        y
sign (y) ?             y
index format (y) ?      n
left of point (0-9) ?   9
right of point (0-9) ?  3
the number :           ***¥1,234.57+
```

```

the number ?          9876.54321
using (y) ?           y
commas (y) ?           n
astarisks (y) ?        n
yen mark (y) ?         n
plus mark (y) ?        n
sign (y) ?             n
index format (y) ?      y
left of point (0-9) ?   9
right of point (0-9) ?  9
the number :           98765432.10000000D-04
```

```

the number ?          1234.5
using (y) ?           n
the number :           1234.5
```

the number ?  
Ok





## 2 E 0 5 H : 単精度型実数の平方根を求める.

アドレス	2 E 0 5 H
機能	単精度型実数の平方根を求める.
レジスタ	A, F, B, C, D, E, H, L

**解 説** フローティング・アキュムレータ (FACC) に格納された単精度型実数の平方根を求め、同じくフローティング・アキュムレータに与えてもどります.

### サンプル

```

;
; --- sqr function test program ---
;
;      ORG 0B900H
;
B900 2138B9 START: LD HL,PROMPT
B903 CD2FB9 CALL DSPMSG
;
B906 CD925F CALL 5F92H ; screen editor
B909 D8 RET C ; stop key, return to basic
;
B90A 23 INC HL
B90B E5 PUSH HL
;
B90C CDB526 CALL 26B5H ; convert into binary code
B90F CD1C22 CALL 221CH ; convert into single precision
B912 CD052E CALL 2E05H ; sqr function
B915 CDD028 CALL 28D0H ; convert into character code
;
B918 EB EX DE,HL
;
B919 2155B9 LD HL,ANS1
B91C CD2FB9 CALL DSPMSG
;
B91F E1 POP HL
B920 CD2FB9 CALL DSPMSG
;
B923 215CB9 LD HL,ANS2
B926 CD2FB9 CALL DSPMSG
;
B929 EB EX DE,HL
B92A CD2FB9 CALL DSPMSG ; display result
;
B92D 18D1 JR START
;
; display message subroutine
;
B92F 7E DSPMSG:LD A,(HL)
B930 A7 AND A
B931 C8 RET Z
B932 CD0D3E CALL 3E0DH
B935 23 INC HL
B936 18F7 JR DSPMSG
;
; message area
;
B938 0D0A7369 PROMPT:DEFB 0DH,0AH,'single precision number ? ',0
B93C 6E676C65
B940 20707265
B944 63697369

```



## 2 E 1 5 H：単精度の指数演算を行なう。

アドレス	2 E 1 5 H
機能	単精度の指数演算を行なう。
レジスタ	A, F, B, C, D, E, H, L

**解 説** レジスタ B C D E に格納された単精度型の実数を第 1 パラメータ (X) とし、フローティング・アキュムレータ (F A C C) に格納された単精度型の実数を第 2 パラメータ (Y) として指数演算を行ない、演算結果 ( $X^Y$ ) をフローティング・アキュムレータ (F A C C) に格納してもどります。

もし、レジスタ B C D E に与える第 1 パラメータが 0 (ゼロ) で、なおかつフローティング・アキュムレータ (F A C C) に与える第 2 パラメータの符号が負 (マイナス) であった場合には、“Division by zero”(／ 0 Error) エラーとなりますがインタプリタは、“／ 0” のメッセージを表示し、単精度型の実数として扱うことのできる最大の数値 ( $1.70141E+38$ ) を指数演算の結果として与えます。

### サンプル

```

;
; --- exponentiation for single precision ---
;
        ORG 0B900H
;
START: LD  HL,PRMPT1
        CALL DSPMSG
        CALL 5F92H          ; screen editor
        RET C               ; stop key, return to basic
        INC HL
        CALL CLRFC          ; clear floating point accumulator
        CALL 26C8H          ; convert into binary code
        CALL 2214H          ; convert into single precision
        CALL 20E8H          ; move from facc to register bcde
        PUSH BC
        PUSH DE
;
        LD  HL,PRMPT2
        CALL DSPMSG
        CALL 5F92H          ; screen editor
        INC HL
        CALL CLRFC          ; clear floating point accumulator
        CALL 26C8H          ; convert into binary code
        CALL 2214H          ; convert into single precision
;
        POP DE
        POP BC
        CALL 2E15H          ; *facc <-- register bcde ^ facc
        CALL 28D0H          ; convert into character code
        PUSH HL
        LD  HL,ANSWER
        CALL DSPMSG
        POP HL
        CALL DSPMSG          ; display result

```

```

;
B93F 18BF      JR    START
;
; clear floating point accumulator subroutine
;
B941 E5      CLRFAC:PUSH HL
B942 210000    LD     HL,0
B945 223DEC    LD     (0EC3DH),HL
B948 223FEC    LD     (0EC3FH),HL
B94B 2241EC    LD     (0EC41H),HL
B94E 2243EC    LD     (0EC43H),HL
B951 E1      POP    HL
B952 C9      RET

;
; display message subroutine
;
B953 7E      DSPMSG:LD     A,(HL)
B954 A7      AND     A
B955 C8      RET     Z
B956 CD0D3E   CALL    3E0DH      ; display a character
B959 23      INC     HL
B95A 18F7     JR     DSPMSG

;
; message area
;
B95C 0D0A0A20 PRMPT1:DEFB 0DH,0AH,0AH,'      x!  ? ',0
B960 20202020
B964 78212020
B968 203F2000
B96C 20202020 PRMPT2:DEFB '      y!  ? ',0
B970 20792120
B974 20203F20
B978 00
B979 7821205E ANSWER:DEFB 'x! ^ y! --> ',0
B97D 20792120
B981 2D2D3E20
B985 00
;

```

a=&hb900:call a

```

x!  ? 0
y!  ? 5
x! ^ y! --> 0

x!  ? 5
y!  ? 0
x! ^ y! --> 1

x!  ? 0
y!  ? -5
/0
x! ^ y! --> 1.70141E+38

x!  ? 3.0001
y!  ? 3.0001
x! ^ y! --> 27.0057

x!  ?
Ok

```

## 2 E 6 E H : e に対する指数関数の値を求める.

アドレス	2 E 6 E H
機 能	e に対する指数関数の値を単精度型実数で求める.
レジスタ	A, F, B, C, D, E, H, L

**解 説** フローティング・アキュムレータ (F A C C) に格納された単精度型実数の, 自然対数の底 e に対する指数関数の値を求め, 同じくフローティング・アキュムレータに与えてもどります.

入力パラメータが, 8 7 . 3 3 6 5 5 より大きい場合には“Overflow” ( O V Error) エラーが発生します.

### サンプル

```

;
; --- exp function test program ---
;
;      ORG 0B900H
;
B900 2138B9 START: LD HL,PROMPT
B903 CD2FB9 CALL DSPMSG
;
B906 CD925F CALL 5F92H ; screen editor
B909 D8 RET C ; stop key, return to basic
;
B90A 23 INC HL
B90B E5 PUSH HL
;
B90C CDB526 CALL 26B5H ; convert into binary code
B90F CD1C22 CALL 221CH ; convert into single precision
B912 CD6E2E CALL 2E6EH ; exp function
B915 CDD028 CALL 28D0H ; convert into character code
;
B918 EB EX DE,HL
;
B919 2155B9 LD HL,ANS1
B91C CD2FB9 CALL DSPMSG
;
B91F E1 POP HL
B920 CD2FB9 CALL DSPMSG
;
B923 215CB9 LD HL,ANS2
B926 CD2FB9 CALL DSPMSG
;
B929 EB EX DE,HL
B92A CD2FB9 CALL DSPMSG ; display result
;
B92D 18D1 JR START
;
; display message subroutine
;
B92F 7E DSPMSG:LD A,(HL)
B930 A7 AND A
B931 C8 RET Z
B932 CD0D3E CALL 3E0DH
B935 23 INC HL
B936 18F7 JR DSPMSG

```

```

;
; message area
;
B938 0D0A7369 PROMPT:DEFB 0DH,0AH,'single precision number ? ',0
B93C 6E676C65
B940 20707265
B944 63697369
B948 6F6E206E
B94C 756D6265
B950 72203F20
B954 00
B955 65787020 ANS1: DEFB 'exp ( ',0
B959 282000
B95C 20292069 ANS2: DEFB ' ) is equal to ',0
B960 73206571
B964 75616C20
B968 746F2000

```

```
defusr=&hb900;a=usr(a)
```

```

single precision number ? 1
exp ( 1 ) is equal to 2.71828
single precision number ? 2
exp ( 2 ) is equal to 7.38906
single precision number ? 3
exp ( 3 ) is equal to 20.0855
single precision number ? 4
exp ( 4 ) is equal to 54.5982
single precision number ? 5
exp ( 5 ) is equal to 148.413
single precision number ? 87
exp ( 87 ) is equal to 6.07602E+37
single precision number ? 88
OV
exp ( 88 ) is equal to 1.70141E+38
single precision number ? abc
exp ( abc ) is equal to 1
single precision number ?
Ok

```



## 2 F 1 A H：単精度実数型の乱数を発生する。

アドレス	2 F 1 A H
------	-----------

機 能	単精度実数型の乱数を発生する。
-----	-----------------

レジスタ	A, F, B, C, D, E, H, L
------	------------------------

解 説	0 以上 1 未満の乱数を単精度型の実数として発生し、フローティング・アキュムレータに与えてもどります。
-----	--

発生する乱数は、N88-BASICのRUNおよびCLEARが実行されるごとに同系列を取り、フローティング・アキュムレータに与える単精度実数型の引数によって次のように異なります。

引数が負の場合には、あたらしい乱数系列を設定します。

引数が0の場合には、1つ前に発生した乱数の値をとります。

引数が正の場合には、次の乱数を発生します。

# サンプル

```

;
; --- display random numbers ---
;
;       ORG 0B900H
;
B900 2133B9      LD HL,TITLE
B903 CD2AB9      CALL DSPMSG
;
B906 0664      LD B,100          ; set loop counter
;
B908 C5      LOOP: PUSH BC
;
B909 210000      LD HL,0          ;
B90C 223DEC      LD (0EC3DH),HL    ;
B90F 223FEC      LD (0EC3FH),HL    ;
B912 2241EC      LD (0EC41H),HL    ; facc <-- single precision of 1
B915 2651      LD H,81          ;
B917 2243EC      LD (0EC43H),HL    ;
B91A CD5222      CALL 2252H        ;
;
B91D CD1A2F      CALL 2F1AH        ; rnd function
B920 CDD028      CALL 28D0H        ; convert into character code
B923 CD2AB9      CALL DSPMSG      ; display a random number
;
B926 C1      POP BC
B927 10DF      DJNZ LOOP          ; check loop counter
;
B929 C9      RET                  ; return to basic
;
; display message subroutine
;
B92A 7E      DSPMSG:LD A,(HL)
B92B A7      AND A
B92C C8      RET Z
B92D CD0D3E      CALL 3E0DH        ; display a character
B930 23      INC HL
B931 18F7      JR DSPMSG
;
; message area
;
B933 0D0A2D2D  TITLE: DEFB 0DH,0AH,'--- random numbers ---',0DH,0AH,0
B937 2D207261
B93B 6E646F6D
B93F 206E756D
B943 62657273
B947 2D2D2D2D
B94B 0D0A00
;

```

a=&hb900:call a

```

--- random numbers ---
.515813 .356502 .669923 .777707 .357315 .0665066 .819051 .791743 .429127 .82081
.531435 .310511 .209423 .708369 .726243 .0934078 .88245 .712778 .118974 .650003
.0462312 .551492 .0901691 .433297 .321401 .448729 .216281 .0159076 .558512 .593
016 .960547 .368405 .352878 .818298 .0490134 .211928 .144036 .265899 .881169 .15
7237 .13199 .69775 .154114 .643139 .428326 .782207 .0468152 .334085 .782977 .682
061 .880143 .892113 .507562 .655652 .709999 .420419 3.66679E-03 .744425 .760866
.988555 .563593 .257445 .563038 .464049 .267248 .699108 .666886 .18031 9.77931E-
03 .565483 .954732 .715359 .929904 .835917 .961199 .229882 .200387 .0347998 .969
152 .988476 .3917 .0847099 .193131 .600219 .578091 .479964 .745441 .753107 .6693
15 .29843 .607734 .585781 .560146 .701109 .709122 .943154 .977027 .716624 .32510
5 .296994
Ok

```

## 2 F 8 B H : 単精度実数型の余弦(コサイン)を求める。

**アドレス** 2 F 8 B H

**機 能** 単精度実数型の余弦(コサイン)を求める。

**レジスタ** A, F, B, C, D, E, H, L

**解 説** フローティング・アキュムレータ(FACC)に格納された単精度型実数の単位をラジアンとした場合の三角関数コサイン(余弦)の値を求め、同じくフローティング・アキュムレータに与えてもどります。

### サンプル

```

;
; --- cos function test program ---
;
        ORG 0B900H
;
B900 2148B9      START: LD  HL,PROMPT
B903 CD42B9      CALL DSPMSG
;
B906 CD925F      CALL 5F92H          ; screen editor
B909 D8          RET C              ; stop key, return to basic
;
B90A 23          INC HL
B90B E5          PUSH HL
;
B90C 110000      LD  DE,0
B90F ED533DEC    LD  (0EC3DH),DE
B913 ED533FEC    LD  (0EC3FH),DE
B917 ED5341EC    LD  (0EC41H),DE
B91B ED5343EC    LD  (0EC43H),DE
;
B91F CDC826      CALL 26C8H          ; convert into binary code
B922 CD1422      CALL 2214H          ; convert into single precision
B925 CD8B2F      CALL 2F8BH          ; cos function
B928 CD0D28      CALL 28D0H          ; convert into character code
;
B92B EB          EX  DE,HL
;
B92C 2168B9      LD  HL,ANS1
B92F CD42B9      CALL DSPMSG
;
B932 E1          POP HL
B933 CD42B9      CALL DSPMSG
;
B936 216FB9      LD  HL,ANS2
B939 CD42B9      CALL DSPMSG
;
B93C EB          EX  DE,HL
B93D CD42B9      CALL DSPMSG          ; display result
;
B940 18BE        JR  START
;
; display message subroutine
;
B942 7E          DSPMSG:LD  A,(HL)
B943 A7          AND  A
B944 C8          RET  Z
B945 CD0D3E      CALL 3E0DH          ; display a character
B948 23          INC  HL
B949 18F7        JR  DSPMSG

```

```

;
; message area
;
B94B 0D0A7369 PROMPT:DEFB 0DH,0AH,'single precision number ? ',0
B94F 6E676C65
B953 20707265
B957 63697369
B95B 6F6E206E
B95F 756D6265
B963 72203F20
B967 00
B968 636F7320 ANS1: DEFB 'cos ( ',0
B96C 282000
B96F 20292069 ANS2: DEFB ' ) is equal to ',0
B973 73206571
B977 75616C20
B97B 746F2000
;

```

```

a=&hb900:call a

```

```

single precision number ? 1
cos ( 1 ) is equal to .540302
single precision number ? 1.1
cos ( 1.1 ) is equal to .453596
single precision number ? 1.2
cos ( 1.2 ) is equal to .362358
single precision number ? 1.3
cos ( 1.3 ) is equal to .267499
single precision number ? 1.4
cos ( 1.4 ) is equal to .169967
single precision number ? 1.5
cos ( 1.5 ) is equal to .0707371
single precision number ? 1.6
cos ( 1.6 ) is equal to -.0291995
single precision number ? 1.7
cos ( 1.7 ) is equal to -.128845
single precision number ?
Ok
print tab(24);cos(1.7)
Ok

```

## 2 F 9 1 H：単精度型実数の正弦(サイン)を求める。

**アドレス** 2 F 9 1 H

**機 能** 単精度型実数の正弦(サイン)を求める。

**レジスタ** A, F, B, C, D, E, H, L

**解 説** フローティング・アキュムレータ (F A C C) に格納された単精度型実数の単位をラジアンとした場合の三角関数サインの値を求め、同じくフローティング・アキュムレータに与えてもどります。

### サンプル

```

;
; --- sin function test program ---
;
; ORG 0B900H
B900 214BB9 START: LD HL,PROMPT
B903 CD42B9 CALL DSPMSG
;
B906 CD925F CALL 5F92H ; screen editor
B909 D8 RET C ; stop key, return to basic
;
B90A 23 INC HL
B90B E5 PUSH HL
;
B90C 110000 LD DE,0 ;
B90F ED533DEC LD (0EC3DH),DE ;
B913 ED533FEC LD (0EC3FH),DE ; clear floating point accumulator
B917 ED5341EC LD (0EC41H),DE ;
B91B ED5343EC LD (0EC43H),DE ;
;
B91F CDC826 CALL 26C8H ; convert into binary code
B922 CD1422 CALL 2214H ; convert into single precision
B925 CD912F CALL 2F91H ; sin function
B928 CD0028 CALL 28D0H ; convert into character code
;
B92B EB EX DE,HL
;
B92C 2168B9 LD HL,ANS1
B92F CD42B9 CALL DSPMSG
;
B932 E1 POP HL
B933 CD42B9 CALL DSPMSG
;
B936 216FB9 LD HL,ANS2
B939 CD42B9 CALL DSPMSG
;
B93C EB EX DE,HL
B93D CD42B9 CALL DSPMSG ; display result
;
B940 18BE JR START
;
; display message subroutine
;
B942 7E DSPMSG:LD A,(HL)
B943 A7 AND A
B944 C8 RET Z
B945 CD003E CALL 3E0DH ; display a character
B948 23 INC HL
B949 18F7 JR DSPMSG

```

```

;
; message area
;
B94B 0D0A7369 PROMPT:DEFB 0DH,0AH,'single precision number ? ',0
B94F 6E676C65
B953 20707265
B957 63697369
B95B 6F6E206E
B95F 756D6265
B963 72203F20
B967 00
B968 73696E20 ANS1: DEFB 'sin ( ',0
B96C 282000
B96F 20292069 ANS2: DEFB ' ) is equal to ',0
B973 73206571
B977 75616C20
B97B 746F2000
;

```

```

a=&hb900:call a

```

```

single precision number ? .1
sin ( .1 ) is equal to .0998334
single precision number ? .2
sin ( .2 ) is equal to .198669
single precision number ? .3
sin ( .3 ) is equal to .29552
single precision number ? .4
sin ( .4 ) is equal to .389418
single precision number ? .5
sin ( .5 ) is equal to .479426
single precision number ? 6
sin ( 6 ) is equal to -.279416
single precision number ? 7
sin ( 7 ) is equal to .656987
single precision number ? 8
sin ( 8 ) is equal to .989358
single precision number ? 9
sin ( 9 ) is equal to .412119
single precision number ? 0
sin ( 0 ) is equal to 0
Ok

```

### 302CH：正接値（タンジェント）を求める。

**アドレス** 302CH

**機能** 単精度型実数の正接値（タンジェント）を求める。

**レジスタ** A, F, B, C, D, E, H, L

**解説** フローティング・アキュムレータ（FACC）に格納された単精度型実数の単位をラジアンとした場合の三角関数タンジェント（正接値）の値を求め、同じくフローティング・アキュムレータに与えてもどります。

#### サンプル

```

;
; --- tan function test program ---
;
; ORG 0B900H
B900 214BB9 START: LD HL,PROMPT
B903 CD42B9 CALL DSPMSG
;
B906 CD925F CALL 5F92H ; screen editor
B909 D8 RET C ; stop key, return to basic
;
B90A 23 INC HL
B90B E5 PUSH HL
;
B90C 110000 LD DE,0 ;
B90F ED533DEC LD (0EC3DH),DE ;
B913 ED533FEC LD (0EC3FH),DE ;
B917 ED5341EC LD (0EC41H),DE ; clear floating point accumulator
B91B ED5343EC LD (0EC43H),DE ;
;
B91F CDC826 CALL 26C8H ; convert into binary code
B922 CD1422 CALL 2214H ; convert into single precision
B925 CD2C30 CALL 302CH ; tan function
B928 CD0028 CALL 28D0H ; convert into character code
;
B92B EB EX DE,HL
;
B92C 2168B9 LD HL,ANS1
B92F CD42B9 CALL DSPMSG
;
B932 E1 POP HL
B933 CD42B9 CALL DSPMSG
;
B936 216FB9 LD HL,ANS2
B939 CD42B9 CALL DSPMSG
;
B93C EB EX DE,HL
B93D CD42B9 CALL DSPMSG ; display result
;
B940 18BE JR START
;
; display message subroutine
DSPMSG:LD A,(HL)
B942 7E AND A
B943 A7 RET Z
B944 C8
B945 CD0D3E CALL 3E0DH ; display a character
B948 23 INC HL
B949 18F7 JR DSPMSG

```



```

;
; message area
;
B94B 0D0A7369 PROMPT:DEFB 0DH,0AH,'single precision number ? ',0
B94F 6E676C65
B953 20707265
B957 63697369
B95B 6F6E206E
B95F 756D6265
B963 72203F20
B967 00
B968 74616E20 ANS1: DEFB 'tan ( ',0
B96C 282000
B96F 20292069 ANS2: DEFB ' ) is equal to ',0
B973 73206571
B977 75616C20
B97B 746F2000
;

```

```

a=&hb900:call a

```

```

single precision number ? 0.001
tan ( 0.001 ) is equal to .001
single precision number ? 0.01
tan ( 0.01 ) is equal to .0100003
single precision number ? 0.1
tan ( 0.1 ) is equal to .100335
single precision number ? 1
tan ( 1 ) is equal to 1.55741
single precision number ? 10
tan ( 10 ) is equal to .64836
single precision number ? 100
tan ( 100 ) is equal to -.587219
single precision number ? 1000
tan ( 1000 ) is equal to 1.47022
single precision number ? 10000
tan ( 10000 ) is equal to .320178
single precision number ? 100000
tan ( 100000 ) is equal to -.0368412
single precision number ?
Ok

```

### 3 5 8 3 H : キーボード 1 文字入力待ちを行なう。

アドレス	3 5 8 3 H
機能	キーボード 1 文字入力待ちを行なう。
レジスタ	A, F

**解 説** キーボードからの入力が認められるまで待ち，入力が有った場合には，入力されたコードをアキュムレータに入れてもどります。

#### サンプル

```

; --- input and display character code ---
;
;      ORG 0B900H
;
B900 CD8335 LOOP: CALL 3583H      ;*input character from keyboard
B903 FE03    CP 03H          ; check break code
B905 C8      RET Z           ; return to basic
;
B906 CD11B9    CALL DSPACC      ; display character code
;
B909 3E20      LD A,' '
B90B CD0D3E    CALL 3E0DH
;
B90E C300B9    JP LOOP
;
; display accumulator hexa decimal subroutine
;
;      destroys : register a,f,b
;
B911 47      DSPACC:LD B,A
B912 0F      RRCA
B913 0F      RRCA
B914 0F      RRCA
B915 0F      RRCA
B916 CD1AB9    CALL HALF
B919 78      LD A,B
;
B91A E60F      HALF: AND 0FH
B91C FE0A      CP 10
B91E 3802      JR C,NUMBER
B920 C607      ADD A,7
;
B922 C630      NUMBER:ADD A,30H
B924 CD0D3E    CALL 3E0DH      ; display a character
B927 C9      RET
;

a=&hb900:call a
64 77 76 75 74 72 71 70 6F 6E 6D 6C 6A 69 72 75 6E 0D 0C 2F 2D 36 35 3D 34 31
3D 33 36 35 31 36 35 36 C9 BC DA C4 CA C9 C4 C1 DA C9 CA C4 7F 7F 12 14
Ok

```

### 3 5 C 2 H : ストップ・キーのチェックを行う。

アドレス	3 5 C 2 H
------	-----------

機 能	ストップ・キーのチェックを行なう。
-----	-------------------

レジスタ	A, F
------	------

解 説	ストップ・キーの入力が認められればキャリ・フラグ(CY)を1にセットし、そうでなければキャリ・フラグ(CY)を0にリセットしてもどります。
-----	---

具体的には、E 6 C A H 番地のデータを調べて、0 3 H (CTRL - C) または 0 4 H (CTRL - D) であればキャリ・フラグ(CY)をセットしています。

# サンプル

```

;
; --- check stop key test program ---
;
        ORG 0B900H
;
B900 2122B9 LOOP: LD HL,MESAGE
B903 CD19B9 CALL DSPMSG
B906 CD0FB9 CALL WAIT ; wait a moment
B909 CDC235 CALL 35C2H ;*check stop key
B90C 30F2 JR NC,LOOP
;
B90E C9 RET ; return to basic
;
; wait subroutine
;
B90F 0EFF WAIT: LD C,0FFH
B911 06FF WAIT1: LD B,0FFH
B913 10FE WAIT2: DJNZ WAIT2
B915 00 DEC C
B916 20F9 JR NZ,WAIT1
B918 C9 RET
;
; display message subroutine
;
B919 7E DSPMSG:LD A,(HL)
B91A A7 AND A
B91B C8 RET Z
B91C CD0D3E CALL 3E0DH ; display a character
B91F 23 INC HL
B920 18F7 JR DSPMSG
;
; message area
;
B922 68697420 MESSAGE:DEFB 'hit stop key please !!',0DH,0AH,0
B926 73746F70
B92A 206B6579
B92E 20706C65
B932 61736520
B936 21210D0A
B93A 00
;

```

```

a=&hb900:call a
hit stop key please !!
hit stop key please !!
hit stop key please !!
^C
Break
Ok

```

### 3 5 C E H : リアル・タイムなキー入力を行なう.

アドレス	3 5 C E H
------	-----------

機 能	リアル・タイムなキー入力を行なう.
-----	-------------------

レジスタ	A, F
------	------

解 説	キーボードをスキャンし、キー入力が認められなければゼロ・フラグ (Z) を 1 にセットし、キー入力が認められればゼロ・フラグ (Z) を 0 にリセットし、アキュムレータに入力コードを格納してもどります.
-----	---

3 5 8 3 H 番地のサブルーチンではキー入力が認められるまで待ち続けますが、このサブルーチンではキー入力の有無にかかわらずもどってくるため、見かけ上の同時処理を行なう事ができます. N88-BASIC 等の INKEY\$ と同等のサブルーチンです.

## サンプル

```

; --- real time key scanning ---
;
; ORG 0B900H
;
B900 2136B9 LOOPPL: LD HL,MESAGE
B903 CD2DB9 CALL DSPMSG
;
B906 0614 LD B,20 ; set loop counter
B908 CDCE35 LOOPS: CALL 35CEH ;*real time key scanning
B90B 200C JR NZ,YESIN
B90D CD24B9 CALL WAIT ; wait a moment
B910 3E3E LD A,'>'
B912 CD0D3E CALL 3E0DH ; display a character
B915 10F1 DJNZ LOOPS
;
B917 18E7 JR LOOPPL
;
B919 F5 YESIN: PUSH AF
B91A 3E20 LD A,' '
B91C CD0D3E CALL 3E0DH
B91F F1 POP AF
B920 CD0D3E CALL 3E0DH ; echo-back a input character
B923 C9 RET ; return to basic
;
; wait subroutine
;
B924 21FF0F WAIT: LD HL,0FFFH
B927 2B WAIT1: DEC HL
B928 7C LD A,H
B929 B5 OR L
B92A 20FB JR NZ,WAIT1
B92C C9 RET
;
; display message subroutine
;
B92D 7E DSPMSG:LD A,(HL)
B92E A7 AND A
B92F C8 RET Z
B930 CD0D3E CALL 3E0DH ; display a character
B933 23 INC HL
B934 18F7 JR DSPMSG
;
; message area
;
B936 0D0A6869 MESSAGE:DEFB 0DH,0AH,'hit any key ',0
B93A 7420616E
B93E 79206B65
B942 792000
;

```

```
a=&hb900:call a
```

[illegible]

### 3 6 9 A H : ディスクと 1 セクタの入出力を行なう。

**アドレス**    3 6 9 A H

**機    能**    ディスクと 1 セクタの入出力を行なう。

**レジスタ**    A, F, D, E, H, L

**解    説**    ディスク入出力のための最も基本的でかつ有効なサブルーチンで、N-BASICの 0 0 9 9 H 番地に相当するものです。

入力パラメータは次のとおりで、いずれも正確に与えてください。

EC 8 5 H 番地    ←    ドライブ・ナンバ (0 から)

EF 5 D H 番地    ←    ドライブのタイプ (0 ~ 3)

レジスタ B        ←    トラック・ナンバ (0 から)

レジスタ C        ←    セクタ・ナンバ (1 から)

レジスタ H L      ←    データ格納メモリのトップ・アドレス

ライト、リード、ベリファイの指定はフラグによって行ないます。

ライトの場合、キャリ・フラグ (CY) をセットします。

リードの場合、ゼロ・フラグ (Z) をリセットします。

ベリファイの場合、ゼロ・フラグ (Z) をセットします。

エラーが発生した場合にはキャリ・フラグ (CY) をセットして、そうで無ければリセットしてもどります。ただしベリファイの指定はリード・アフタ・ライトとは根本的に異なります。

また指定したドライブが両面仕様の場合にはトラック・ナンバが奇数の場合には表面を、偶数の場合には裏面をアクセスするためサーフェイスの指定は必要ありません。

#### サンプル

```

; --- control disk ---
;
;          ORG 0B900H
;
B900 217DB9      LD    HL,MSGDR
B903 CD74B9      CALL  DSPMSG
B906 CD58B9      CALL  INDEC
B909 3D          DEC   A
B90A 3285EC      LD    (0EC85H),A          ; set drive number
;
B90D CDC83D      CALL  3DCBH              ; set type of drive
;
B910 2189B9      LD    HL,MSGTR
B913 CD74B9      CALL  DSPMSG
B916 CD58B9      CALL  INDEC
B919 47          LD    B,A                ; set track number
```



```

B91A 2195B9      ; LD HL,MSGSEC
B91D CD74B9      CALL DSPMSG
B920 CD58B9      CALL INDEC
B923 4F          LD C,A          ; set sector number

B924 21A1B9      ; LD HL,MSGWRV
B927 CD74B9      CALL DSPMSG
B92A CD8335      CALL 3583H      ; input a character from keyboard
B92D CD0D3E      CALL 3E0DH      ; echo-back a character

B930 FE77      ; CP 'w'
B932 2813      JR Z,WRITE
B934 FE72      CP 'r'
B936 2812      JR Z,READ
B938 FE76      CP 'v'
B93A 2812      JR Z,VERIFY
B93C 1813      JR ERROR

;
B93E 2100BA      CTRLDS:LD HL,BUFFER      ; set top address of buffer
B941 CD9A36      CALL 369AH      ;*control disk
B944 380B      JR C,ERROR
B946 C9          RET          ; return to basic

B947 37          ; WRITE: SCF
B948 18F4      JR CTRLDS      ; set carry flag

B94A AF          ; READ: XOR A          ; reset carry flag
B94B 3C          INC A          ; reset zero flag
B94C 18F0      JR CTRLDS

;
B94E AF          ; VERIFY:XOR A          ; reset carry and set zero flag
B94F 18ED      JR CTRLDS

;
B951 21BCB9      ERROR:LD HL,MSGERR      ;
B954 CD74B9      CALL DSPMSG      ; error, return to basic
B957 C9          RET          ;

; input decimal number subroutine
;
; outputs : input number in accumulator
; destroys : register a,f,e
;
B958 1E00      INDEC:LD E,0          ; initialize input number
B95A 3E30      LD A,'0'          ; initialize input character

;
B95C F5          INLOOP:PUSH AF          ;
B95D 7B          LD A,E          ;
B95E 87          ADD A,A          ;
B95F 87          ADD A,A          ;
B960 83          ADD A,E          ;
B961 87          ADD A,A          ; e <-- e * 10 + input number
B962 5F          LD E,A          ;
B963 F1          POP AF          ;
B964 D630      SUB '0'          ;
B966 83          ADD A,E          ;
B967 5F          LD E,A          ;

;
B968 CD8335      CALL 3583H      ; input a character from keyboard
B96B CD0D3E      CALL 3E0DH      ; echo-back a character

;
B96E FE0D      CP 0DH
B970 20EA      JR NZ,INLOOP

;
B972 7B          LD A,E
B973 C9          RET

; display message subroutine
;
B974 7E          DSPMSG:LD A,(HL)
B975 A7          AND A
B976 C8          RET Z
B977 CD0D3E      CALL 3E0DH      ; display a character
B97A 23          INC HL
B97B 18F7      JR DSPMSG

```

```

;
; message area
;
B97D 0D0A6472 MSGDR: DEFB 0DH,0AH,'drive ? ',0
B981 69766520
B985 203F2000
B989 0D0A7472 MSGTR: DEFB 0DH,0AH,'track ? ',0
B98D 61636B20
B991 203F2000
B995 0D0A7365 MSGSEC:DEFB 0DH,0AH,'sector ? ',0
B999 63746F72
B99D 203F2000
B9A1 0D0A7772 MSGWRV:DEFB 0DH,0AH,'write, read or verify ? ',0
B9A5 6974652C
B9A9 20726561
B9AD 64206F72
B9B1 20766572
B9B5 69667920
B9B9 3F2000
B9BC 0D0A6572 MSGERR:DEFB 0DH,0AH,'error !!',07H,0
B9C0 726F7220
B9C4 21210700
;
; equate buffer area
;
BA00 BUFFER:EQU 0BA00H
;

```

a=&hb900:call a

```

drive ? 1
track ? 0
sector ? 1
write, read or verify ? r
Ok
mon

```

hJdba00,baff

```

BA00 21 00 84 AF 32 B4 EC 01 02 00 11 11 2D 3A 5D EF
BA10 B7 20 04 06 02 1E 1B AF 3C E5 D5 C5 CD 9A 36 C1
BA20 D1 E1 30 08 3A B4 EC FE 03 20 EC C9 AF 32 B4 EC
BA30 15 2B 0B 0C 24 7B B9 20 DE 04 0E 01 18 D9 CD 00
BA40 84 C3 00 88 22 E8 E7 C9 06 28 21 60 C0 5E 23 56
BA50 23 4E 23 7E 23 EB 36 C3 23 71 23 77 EB 10 EE C9
BA60 DC EC AC D9 C4 EC A2 D0 C1 EC DE D9 D0 EC 14 D4
BA70 AE ED 8E D9 E8 EC A4 D4 B1 ED 4A CE EE EC EB D4
BA80 F1 EC 6F D9 F4 EC 82 D9 6F ED FE CF AB ED E1 CE
BA90 FF ED 76 CE BB EC 4B C3 D3 EC D8 ED D6 EC D8 ED
BAA0 9F ED D8 ED A2 ED D8 ED A2 ED A5 ED A8 ED D8 ED
BAB0 74 EE D8 ED 77 EE D8 ED 7A EE E0 D8 7D EE 22 C2
BAC0 80 EE CC DA 83 EE F6 DA 86 EE A5 DE 89 EE 6D DB
BAD0 8C EE 56 D2 8F EE 96 CE 92 EE 8A D0 95 EE 08 D5
BAE0 98 EE 36 D5 9B EE 49 D3 9E EE 44 D3 8C EE 00 C1
BAF0 BF EE 23 DA C2 EE E7 D9 C5 EE A2 DE C8 EE EC C2

```

hJ^b

Ok

a=&hb900:call a

```

drive ? 1
track ? 0
sector ? 1
write, read or verify ? x
error !!
Ok

```

```

! 22I● -;J\
+ ッくハナハ6チ
△H0 :I● ●ノ22I●
( $C" ルハ
#チI'△ノ(1'タ^#V
#N#~#△6チ#q#ル●ノ
ワ●#ルト●ミチ●#ミ●ナ
ヨC#ル△●、ナアQ#ル●#ナ
門●●#日●●●●マオド#
Q●ネサ●●テモ●リQ●リO
ノリQ'●リQ'●O' Q'●リQ
てノリQ'●リQ'●=□)ノ'ツ
ノフレ●●カレ●'・! /m0
■ノメ+ノ|ナノ|ミーノユ
rノ6ユ+ノ|モノDモシノチ
リノ#レリノノルナノ'ネノ●リ

```

## 36E2H: PC-8031のイニシアライズを行なう。

**アドレス** 36E2H

**機能** PC-8031のイニシアライズを行なう。

**レジスタ** A, F, D

**解説** インテリジェント・タイプのミニ・フロッピー・ディスク・ユニット（PC-8031またはPC-8031-2W同等品）とのインタフェースをイニシアライズし、イニシアライズ・コマンドの送信後、接続されているドライバの数をアキュムレータに格納してもどります。

ディスク・ユニットが未接続の場合にはアキュムレータに00Hが与えられます。

### サンプル

```

;
; --- count connected drives of pc-8031 ---
;
;      ORG 0B900H
;
B900 CDE236      CALL 36E2H          ; send initialize command to 8031
;
B903 C630        ADD A,'0'          ; convert into a character
B905 CD0D3E      CALL 3E0DH          ; display a character
;
B908 2118B9      LD HL,MESAGE
B90B CD0FB9      CALL DSPMSG
;
B90E C9          RET                ; return to basic
;
; display message subroutine
;
B90F 7E          DSPMSG:LD A,(HL)
B910 A7          AND A
B911 C8          RET Z
B912 CD0D3E      CALL 3E0DH          ; display a character
B915 23          INC HL
B916 18F7        JR DSPMSG
;
; message area
;
B918 20647269    MESSAGE:DEFB ' drives are connected !!',0DH,0AH,0
B91C 76657320
B920 61726520
B924 636F6E6E
B928 65637465
B92C 64202121
B930 0D0A00
;

```

```

a=&hb900:call a
2 drives are connected !!
Ok

```

## 37C9H: PC-8031へコマンドを送信する。

アドレス	37C9H
機能	PC-8031へコマンドを送信する。
レジスタ	A, F

**解 説** アキュムレータに格納された1バイトのデータを、インテリジェント・タイプのミニ・フロッピー・ディスク・ユニット (PC-8031またはPC-8031-2W等) に、コマンドとして送信します。

送信するデータがコマンドであるかデータであるかの区別は、ハンドシェイクの際にATN信号をセットするかりセットするかで行ないます。

### サンプル

```

;
; --- initialize pc-8031 ---
;
;      ORG 0B900H
;
B900 3E00      LD  A,0                ; set initialize command
B902 CDC937    CALL 37C9H            ; send command byte to pc-8031
;
B905 2115B9    LD  HL,MESAGE
B908 C00CB9    CALL DSPMSG
;
B90B C9        RET                  ; return to basic
;
; display message subroutine
;
B90C 7E        DSPMSG:LD  A,(HL)
B90D A7        AND  A
B90E C8        RET  Z
B90F CD0D3E    CALL 3E0DH            ; display a character
B912 23        INC  HL
B913 18F7      JR   DSPMSG
;
; message area
;
B915 696E6974  MESSAGE:DEFB 'initialized !!',0DH,0AH,0
B919 69616C69
B91D 7A656420
B921 21210D0A
B925 00
;

```

```

a=&hb900:call a
initialized !!
Ok

```

## 37D2H: PC-8031ヘータを送信する.

アドレス	37D2H
------	-------

機能	PC-8031ヘータを送信する.
----	------------------

レジスタ	F
------	---

解説	インテリジェント・タイプのミニ・フロッピ・ディスク・ユニット (PC-8031またはPC-8031-2W) へ, アキュムレータのデータ1バイトを送信してもどります.
----	---

エラー時には, キャリ・フラグ (CY) をセットします.

# サンプル

```

;
; --- set up only drive 2 for single surface ---
;
;      ORG 0B900H
B900 3E17      LD  A,17H
B902 CDC937    CALL 37C9H      ; output a command to disk
;
B905 3E0D      LD  A,0DH
B907 CDD237    CALL 37D2H      ; output a data to disk
;
B90A 3E02      LD  A,02H
B90C 3265EF    LD  (0EF65H),A
;
B90F 211FB9    LD  HL,MESAGE
B912 CD16B9    CALL DSPMSG
;
B915 C9        RET      ; return to basic
;
; display message subroutine
;
B916 7E        DSPMSG:LD  A,(HL)
B917 A7        AND  A
B918 C8        RET  Z
B919 CD0D3E    CALL 3E0DH      ; display a character
B91C 23        INC  HL
B91D 18F7      JR   DSPMSG
;
; message area
;
B91F 0D0A      MESSAGE:DEFB 0DH,0AH
B921 64726976  DEFM 'drive 1 for double surface !'
B925 65203120
B929 666F7220
B92D 646F7562
B931 6C652073
B935 75726661
B939 63652021
B93D 0D0A      DEFB 0DH,0AH
B93F 64726976  DEFM 'drive 2 for single surface !!'
B943 65203220
B947 666F7220
B94B 73696E67
B94F 6C652073
B953 75726661
B957 63652021
B95B 21
B95C 0D0A00    DEFB 0DH,0AH,0

```

a=&hb900:call a

drive 1 for double surface !  
drive 2 for single surface !!  
Ok

3 8 4 7 H : P C - 8 0 3 1 からのデータ受信を行なう。

アドレス	3 8 4 7 H
------	-----------

機能	P C - 8 0 3 1 からのデータ受信を行なう。
----	-----------------------------

レジスタ	A, F
------	------

解説	インテリジェント・タイプのミニ・フロッピ・ディスク・ユニット (P C - 8 0 3 1 または P C - 8 0 3 1 - 2 W 等) から 1 バイトのデータを受け 取り、アキュムレータに格納してもどります。
----	--

ただし、一定時間以上待ってもデータが送信されて来ない場合には、データを受信せずにキャリ・フラグ (C Y) を 1 にセットするのみでもどります。



# サンプル

```

;
; --- check the unit pc-8031 or pc-8031-2w ---
;
; ORG 0B900H
;
B900 3E0B LD A,0BH ; transmit command in accumulator
B902 CDC937 CALL 37C9H ; send command byte to pc-8031
;
B905 3E07 LD A,07H ; address high byte
B907 CDD237 CALL 37D2H ; send a data to pc-8031
B90A 3EEF LD A,0EFH ; address low byte
B90C CDD237 CALL 37D2H ; send a data to pc-8031
;
B90F 3E00 LD A,00H ; number of datum high byte
B911 CDD237 CALL 37D2H ; send a data to pc-8031
B914 3E01 LD A,01H ; number of datum low byte
B916 CDD237 CALL 37D2H ; send a data to pc-8031
;
B919 CD4738 CALL 3847H ; receive a data from pc-8031
B91C 2F CPL ;
B91D E6F0 AND 0F0H ;
B91F FE10 CP 10H ;
B921 2007 JR NZ,NOT2W ;
;
B923 213AB9 YES2W: LD HL,MSG2W
B926 CD31B9 CALL DSPMSG
B929 C9 RET ; return to basic
;
B92A 2157B9 NOT2W: LD HL,MSG1V
B92D CD31B9 CALL DSPMSG
B930 C9 RET ; return to basic
;
; display message subroutine
;
B931 7E DSPMSG:LD A,(HL)
B932 A7 AND A
B933 C8 RET Z
B934 CD0D3E CALL 3E0DH ; display a character
B937 23 INC HL
B938 18F7 JR DSPMSG
;
; message area
;
B93A 70632D38 MSG2W: DEFB 'pc-8031-2w is connected !!',0DH,0AH,0
B93E 3033312D
B942 32772069
B946 7320636F
B94A 6E6E6563
B94E 74656420
B952 21210D0A
B956 00
B957 6E6F7420 MSG1V: DEFB 'not pc-8031-2w !!',0DH,0AH,0
B95B 70632D38
B95F 3033312D
B963 32772021
B967 210D0A00
;

```

```

a=&hb900:call a
pc-8031-2w is connected !!
Ok

```

### 3DCBH: ディスク・ドライブのタイプを調べる.

**アドレス** 3DCBH

**機能** ディスク・ドライブのタイプを調べる.

**レジスタ** A, F, H, L

**解説** アキュムレータのドライブ・ナンバ(0から)に対応するドライブのタイプ(種類)を調べ、アキュムレータおよびEF5DH番地に格納してもどります。ドライブのタイプと数値の関係は次のとおりです。

DMA転送方式による8インチの場合0となります。

DMA転送方式による5インチの場合1となります。

インテリジェント・タイプで片面の場合2となります。

インテリジェント・タイプで両面の場合3となります。

N88-BASICではEF64H番地から各ドライブのタイプを格納してあり、3DCBH番地からのサブルーチンでは、このワーク・エリアのデータを取り出しています。

#### サンプル

```

;
; --- check type of drive ---
;
; ORG 0B900H
B900 2146B9 START: LD HL,PROMPT
B903 CD3DB9 CALL DSPMSG
;
B906 CD8335 CALL 3583H ; input a character from keyboard
B909 CD0D3E CALL 3E0DH ; echo-back a character
B90C FE03 CP 03H
B90E C8 RET Z ; stop key, return to basic
B90F FE31 CP '1'
B911 38ED JR C,START ; error, input drive number again
B913 FE39 CP '8'+1
B915 30E9 JR NC,START ; error, input drive number again
;
B917 D630 SUB '0' ; convert into binary code
B919 3D DEC A
;
B91A CDCB3D CALL 3DCBH ;get type of drive
;
B91D A7 AND A
B91E 2005 JR NZ,NOT0
B920 215BB9 LD HL,ANS0
B923 1813 JR DISPLAY
;
B925 3D NOT0: DEC A
B926 2005 JR NZ,NOT1
B928 216DB9 LD HL,ANS1
B92B 180B JR DISPLAY
;
B92D 3D NOT1: DEC A
B92E 2005 JR NZ,NOT2
B930 217FB9 LD HL,ANS2
B933 1803 JR DISPLAY

```

```

B935 219BB9      ;
NOT2: LD HL,ANS3
B938 CD3DB9      ;
DISPLAY:CALL DSPMSG ; display type of drive
B93B 18C3        ;
JR START
;
; display message subroutine
;
B93D 7E          DSPMSG:LD A,(HL)
B93E A7          AND A
B93F C8          RET Z
B940 CD0D3E      CALL 3E0DH ; display a character
B943 23          INC HL
B944 18F7        JR DSPMSG
;
; message area
;
B946 0D0A6E75    PROMPT:DEFB 0DH,0AH,'number of drive ? ',0
B94A 6D626572
B94E 206F6620
B952 64726976
B956 65203F20
B95A 00
B95B 0D0A646D    ANS0: DEFB 0DH,0AH,'dma 8 inches !!',0
B95F 61203820
B963 696E6368
B967 65732021
B96B 2100
B96D 0D0A646D    ANS1: DEFB 0DH,0AH,'dma 5 inches !!',0
B971 61203520
B975 696E6368
B979 65732021
B97D 2100
B97F 0D0A696E    ANS2: DEFB 0DH,0AH,'intelligent single sur !!',0
B983 74656C6C
B987 6967656E
B98B 74207369
B98F 6E676C65
B993 20737572
B997 20212100
B99B 0D0A696E    ANS3: DEFB 0DH,0AH,'intelligent double sur !!',0
B99F 74656C6C
B9A3 6967656E
B9A7 7420646F
B9AB 75626C65
B9AF 20737572
B9B3 20212100
;

```

```
a=&hb900:call a
```

```

number of drive ? 0
number of drive ? 1
intelligent double sur !!
number of drive ? 2
intelligent double sur !!
number of drive ?
Ok

```

### 3DD9H:PC-8031のドライブ・ナンバを求める.

アドレス	3DD9H
機能	PC-8031のドライブ・ナンバを求める.
レジスタ	A, F, H, L

**解 説** DMA転送方式による8インチのドライブ, DMA転送方式による5インチのドライブ, インテリジェント・タイプのドライブ全てを含めたドライブ・ナンバをEC85Hに格納してコールする事によって, アキュムレータにインテリジェント・タイプのみのドライブ・ナンバを与えます.

具体的には, EC85H番地のデータから, DMA方式による8インチのドライブ数(EF60H番地)およびDMA方式による5インチのドライブ数(EF61H番地)を減じたデータをアキュムレータに与えます.

# サンプル

```

;
; --- calculate drive number of pc-8031 ---
;
        ORG 0B900H
;
B900 2133B9      LD HL,PROMPT
B903 CD2AB9      CALL DSPMSG
;
B906 CD8335      CALL 3583H          ; input a character from keyboard
B909 FE31        CP '1'
B90B DB          RET C              ; error, return to basic
B90C FE39        CP '8'+1
B90E D0          RET NC             ; error, return to basic
;
B90F CD0D3E      CALL 3E0DH          ; echo-back a character
B912 D630        SUB '0'            ; convert into binary code
B914 3D          DEC A
B915 3285EC      LD (0EC85H),A
;
B918 CDD93D      CALL 3DD9H          ;*calculate drive number of 8031
;
B91B 3C          INC A
B91C C630        ADD A,'0'          ; convert into character code
B91E 47          LD B,A
;
B91F 2150B9      LD HL,ANSWER
B922 CD2AB9      CALL DSPMSG
;
B925 78          LD A,B
B926 CD0D3E      CALL 3E0DH          ; display the result
;
B929 C9          RET                ; return to basic
;
; display message subroutine
;
B92A 7E          DSPMSG:LD A,(HL)
B92B A7          AND A
B92C C8          RET Z
B92D CD0D3E      CALL 3E0DH          ; display a character
B930 23          INC HL
B931 18F7        JR DSPMSG
;
; message area
;
B933 0D0A6472    PROMPT:DEFB 0DH,0AH,'drive number from one ? ',0
B937 69766520
B93B 6E756D62
B93F 65722066
B943 726F6D20
B947 6F6E6520
B94B 20203F20
B94F 00
B950 0D0A6472    ANSWER:DEFB 0DH,0AH,'drive number of pc-8031 : ',0
B954 69766520
B958 6E756D62
B95C 6572206F
B960 66207063
B964 2D383033
B968 31203A20
B96C 00
;

```

a=&hb900:call a

drive number from one ? 1  
drive number of pc-8031 : 1  
Ok

### 3E0DH: CRTスクリーン1バイト出力を行なう.

アドレス	3E0DH
機能	CRTスクリーン1バイト出力を行なう.
レジスタ	全て保存

**解説** アキュムレータのデータ1バイトを、CRTスクリーン上のカーソル位置に出力します.

データが20H以上の場合には通常のキャラクタとしてスクリーン上に表示しますが、20H未満のコントロール・コードであればCRTスクリーンのクリアやカーソルの移動等、コントロール・コードとしての動作を行ないます.

#### サンプル

```

;
; --- display characters ---
;
;          ORG  0B900H
B900 3E20      ;          LD   A,20H
;
B902 CD0D3E   LOOP: CALL 3E0DH          ;*output a character to crt screen
B905 3C        INC  A
B906 FE00      CP   OFFH+1
B908 20F8      JR   NZ,LOOP
;
B90A C9        RET                      ; return to basic
;

```

```

a=&hb900:call a
!`#%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
~~~~~■■■■■+H F-I l n p u 。「」・ヲアイウエオ+13ツァイウエオ+カキクコ+サシセソ
タチツテニスネノハヒフヘホ+ミムメモ+ヤヨ+リルレロ+ン*°=≡≡ ▲▼◆♦★●○/×/計年月日時分秒
Ok

```

### 3 E C 0 H : 内蔵ブザーをコントロールする.

アドレス 3 E C 0 H

機能 内蔵ブザーをコントロールする.

レジスタ A, F

解説 アキュムレータのデータが 0 の場合には内蔵ブザーを OFF (BEEP 0) に, アキュムレータのデータが 0 以外の場合には内蔵ブザーを ON (BEEP 1) にします.

#### サンプル

```

;
; --- american patrol siren ---
;
;      ORG 0B900H
B900 160A      LD D,10
;
B902 0E01      LOOP1: LD C,1
;
B904 41        LOOP2: LD B,C
B905 CD1CB9    CALL BEEP1
B908 10FE      THIS1: DJNZ THIS1
;
B90A 41        LD B,C
B90B CD17B9    CALL BEEP0
B90E 10FE      THIS2: DJNZ THIS2
;
B910 0C        INC C
B911 20F1      JR NZ,LOOP2
;
B913 15        DEC D
B914 20EC      JR NZ,LOOP1
;
B916 C9        RET ; return to basic
;
; buzzer off subroutine
;
B917 AF      BEEP0: XOR A
B918 CDC03E  CALL 3EC0H ; control buzzer
B91B C9      RET
;
; buzzer on subroutine
;
B91C 3E01      BEEP1: LD A,1
B91E CDC03E  CALL 3EC0H ; control buzzer
B921 C9      RET
;

```



### 3ED4H: プリンタへの1バイト出力を行なう.

アドレス	3ED4H
機能	プリンタへの1バイト出力を行なう.
レジスタ	全て保存

**解説** アキュムレータのデータ1バイトを、プリンタへ出力します. ただし出力直前に、ストップ・キーのチェック・ルーチンをコールしており、ブレーク・コードの入力が認められた場合にはプリンタへの出力を止めてもどります.

#### サンプル

```

;
; --- output message to printer ---
;
;      ORG 0B900H
;
B900 210DB9      LD    HL,MESAGE
;
B903 7E          LOOP: LD    A,(HL)      ; get a character
B904 FE00        CP     0                ; check end mark
B906 C8          RET    Z                ; return to basic
;
B907 CDD43E      CALL 3ED4H              ;*output a character to printer
B90A 23          INC    HL              ; increment pointer
B90B 18F6        JR     LOOP
;
; message area
;
B90D 6F757470    MESSAGE:DEFB 'output message for printer !',0DH,0
B911 7574206D
B915 65737361
B919 67652066
B91D 6F722070
B921 72696E74
B925 65722021
B929 0000
;

```

output message for printer !

## 4 0 4 7 H : タイマからデータの読み込み行なう。

アドレス	4 0 4 7 H
機能	タイマからデータの読み込みを行なう。
レジスタ	A, F, B, C, D, E, H, L

**解 説** ハード・タイマの情報を読み取り、ワーク・エリア (F 0 0 D H ~ F 0 1 2 番地) に格納します。それぞれのデータはBCDコードに変換し格納しますが、MONTH (F 0 1 1 H 番地) のみは例外で、普通のバイナリ・コードとして格納します。

以下に、タイマ制御用のワーク・エリアを示します。

F 0 0 D H 番地 ← SECOND (秒: BCD コード)  
 F 0 0 E H 番地 ← MINUTE (分: BCD コード)  
 F 0 0 F H 番地 ← HOUR (時: BCD コード)  
 F 0 1 0 H 番地 ← DAY (日: BCD コード)  
 F 0 1 1 H 番地 ← MONTH (月: バイナリ・コード)  
 F 0 1 2 H 番地 ← YEAR (年: BCD コード)

### サンプル

```

;
; --- read from timer ---
;
;      ORG 0B900H
; read from timer section
;
B900 CD4740      CALL 4047H          ;*read from timer
;
; display date section
;
B903 2173B9      LD  HL,ADATE
B906 CD6AB9      CALL DSPMSG
;
B909 2112F0      LD  HL,0F012H      ; end address of date buffer
B90C 0603        LD  B,3
B90E 7E         LDATE: LD  A,(HL)
B90F 2B         DEC  HL
;
B910 48         LD  C,B
B911 0D         DEC  C
B912 0D         DEC  C
B913 2805       JR   Z,MONTH        ; check minute or other
;
B915 CD39B9     CALL BCDCHR          ; convert bcd into 2 characters
B918 1803       JR   SKIP
;
B91A CD4CB9     MONTH: CALL CMONTH   ; convert month into 2 characters
B91D CD5CB9     SKIP:  CALL DSPDE    ; display two characters in reg-de
B920 10EC       DJNZ LDATE
  
```

```

;
; display time section
;
B922 218AB9      LD    HL,ATIME
B925 C06AB9      CALL  DSPMSG

;
B928 210FF0      LD    HL,0F012H-3      ; end address of date buffer
B92B 0603        LD    B,3
B92D 7E          LTIME: LD    A,(HL)
B92E 2B          DEC    HL

;
B92F C039B9      CALL  BCDCHR      ; convert bcd into 2 characters
B932 C05CB9      CALL  DSPDE      ; display two characters in reg-de
B935 1B          DEC    DE
B936 10F5        DJNZ  LTIME

;
B938 C9          RET      ; return to basic

;
; convert bcd code into character code subroutine
;
; inputs   : bcd code in accumulator
; outputs  : two characters in register de
; destroys : register a,f,d,e
;
B939 5F          BCDCHR:LD    E,A
B93A CB3F        SRL    A
B93C CB3F        SRL    A
B93E CB3F        SRL    A
B940 CB3F        SRL    A
B942 C630        ADD    A,'0'
B944 57          LD     D,A

;
B945 7B          LD     A,E
B946 E60F        AND    0FH
B948 C630        ADD    A,'0'
B94A 5F          LD     E,A
B94B C9          RET

;
; convert month into character code subroutine
;
; inputs   : month in accumulator
; outputs  : two characters in register de
; destroys : register a,f,d,e
;
B94C 1630        CMONTH:LD    D,'0'
B94E FE0A        CP     0AH
B950 3004        JR     NC,CMONT1
B952 C630        ADD    A,'0'
B954 5F          LD     E,A
B955 C9          RET

;
B956 1631        CMONT1:LD    D,'1'
B958 C626        ADD    A,'0'-0AH
B95A 5F          LD     E,A
B95B C9          RET

;
; display two characters
;
; destroys : accumulator
;
B95C 7A          DSPDE: LD    A,D
B95D C00D3E      CALL  3E00H      ; display a character
B960 7B          LD    A,E
B961 C00D3E      CALL  3E00H      ; display a character
B964 3E20        LD    A,' '
B966 C00D3E      CALL  3E00H      ; display a space code
B969 C9          RET

;
; display message subroutine
;
; destroys : register a,f,h,l
;

```

```

B96A 7E      DSPMSG:LD    A,(HL)
B96B A7      AND    A
B96C C8      RET    Z
B96D CD0D3E  CALL    3E0DH      ; display a character
B970 23      INC    HL
B971 18F7    JR      DSPMSG

```

```

;
; message area

```

```

B973 0D0A6461 ADATE: DEFB 0DH,0AH,'date - yy mm dd --> ',0
B977 7465202D
B97B 20797920
B97F 6D6D2064
B983 64202D2D
B987 3E2000
B98A 0D0A7469 ATIME: DEFB 0DH,0AH,'time - hh mm ss --> ',0
B98E 6D65202D
B992 20686820
B996 6D6D2073
B99A 73202D2D
B99E 3E2000

```

```

;

```

```

print date$:print time$

```

```

01/12/24

```

```

00:39:27

```

```

Ok

```

```

a=&hb900:call a

```

```

date - yy mm dd --> 01 12 24

```

```

time - hh mm ss --> 00 39 37

```

```

Ok

```

```

date$='82/09/01':time$='00:00:00'

```

```

Ok

```

```

a=&hb900:call a

```

```

date - yy mm dd --> 82 09 01

```

```

time - hh mm ss --> 00 00 08

```

```

Ok

```

## 4 2 8 B H : カーソルを消去する。

アドレス	4 2 8 B H
機 能	カーソルを消去する。
レジスタ	全て保存

**解 説** CRTスクリーン上のカーソル位置に表示すべきカーソルを表示しません。ただし、N88-BASICやモニタ等からカーソル表示ルーチンが実行されたり、CRTコントローラのイニシアライズが行なわれたりした場合には、カーソルの表示が再開されます。

具体的には、CRTコントローラに与えるカーソル・コントロール・コマンドが格納される、E 6 A 8 H番地に8 0 Hを格納する事によって、カーソルの消去を実現しています。

# サンプル

```

;
; --- cursor on and off ---
;
        ORG 0B900H
;
B900 2136B9 LOOP: LD HL,MSGON
B903 CD20B9 CALL DSPMSG
B906 CD9042 CALL 4290H ; display cursor
B909 CD1EB9 CALL WAIT ; wait a moment
;
B90C 2146B9 LD HL,MSGOFF
B90F CD20B9 CALL DSPMSG
B912 CD8B42 CALL 428BH ;*not display cursor
B915 CD1EB9 CALL WAIT ; wait a moment
;
B918 CDC235 CALL 35C2H ; check stop key
B91B 30E3 JR NC,LOOP
;
B91D C9 RET ; return to basic
;
; wait subroutine
;
B91E 3E0F WAIT: LD A,0FH
B920 0EFF WAIT1: LD C,0FFH
B922 06FF WAIT2: LD B,0FFH
B924 10FE WAIT3: DJNZ WAIT3
B926 00 DEC C
B927 20F9 JR NZ,WAIT2
B929 3D DEC A
B92A 20F4 JR NZ,WAIT1
B92C C9 RET
;
; display message subroutine
;
B92D 7E DSPMSG:LD A,(HL)
B92E A7 AND A
B92F C8 RET Z
B930 CD0D3E CALL 3E0DH ; display a character
B933 23 INC HL
B934 18F7 JR DSPMSG
;
; message area
;
B936 0D0A6375 MSGON: DEFB 0DH,0AH,'cursor on !! ',0
B93A 72736F72
B93E 206F6E20
B942 21212000
B946 0D0A6375 MSGOFF:DEFB 0DH,0AH,'cursor off !! ',0
B94A 72736F72
B94E 206F6666
B952 20212120
B956 00
;

```

a=&hb900:call a

```

cursor on !! ■
cursor off !!
cursor on !! ■
cursor off !!
cursor on !! ■
cursor off !!
cursor on !! ■
cursor off !! ^C
Break
Ok

```

## 4 2 9 0 H : カーソルの表示を行なう.

アドレス	4 2 9 0 H
機 能	カーソルの表示を行なう.
レジスタ	全て保存
解 説	C R T スクリーン上のカーソル位置に表示すべきカーソルを表示します.

通常, N<sub>88</sub>-B A S I C等を用いて業務用のプログラム中でキーボードからのデータ入力を行なう場合には, 見かけ上の同時処理を実現するため, およびエラー処理の完璧を期するために, 普通の入力命令をさけて I N K E Y \$ を使用しますが, I N K E Y \$ の実行に際してはカーソルが消去されたままです. このような場合に, カーソルを表示するためにも, カーソル表示ルーチンを利用する事ができます.



## サンプル

```

100 /
110 / input routine for business program
120 /
150 DEFINT A-Z
155 CURSORON=&H4290
160 LENGTH=30
163 STOP ON
164 /
165 *START
170 PRINT
171 PRINT 'input ? ';
200 GOSUB *SUBINPUT
201 IF RESULT$='end' THEN STOP OFF:END
205 PRINT
210 PRINT 'print : ';RESULT$
220 GOTO *START
230 /
250 *SUBINPUT
290 COLUMN=0
300 RESULT$=''
310 PRINT STRING$(LENGTH, '_');STRING$(LENGTH, CHR$(&H1D));
320 /
330 *LOOP
340 CALL CURSORON
350 IN$=INKEY$
360 IF IN$='' THEN *LOOP
365 /
370 IN=ASC(IN$)
380 IF IN= &H8 THEN GOSUB *SUBDEL:GOTO *LOOP
390 IF IN=&H7F THEN GOSUB *SUBDEL:GOTO *LOOP
395 IF IN=&H15 THEN WHILE 0<COLUMN:GOSUB *SUBDEL:WEND
400 IF IN= &HD THEN RETURN
410 IF IN<&H20 THEN *LOOP
411 /
415 IF COLUMN=LENGTH THEN *LOOP
420 COLUMN=COLUMN+1
430 RESULT$=RESULT$+IN$
440 PRINT IN$;
450 GOTO *LOOP
500 /
510 *SUBDEL
520 IF COLUMN=0 THEN RETURN
530 COLUMN=COLUMN-1
540 RESULT$=LEFT$(RESULT$,COLUMN)
550 PRINT CHR$(&H1D); '_';CHR$(&H1D);
560 RETURN

```

run

```

input ? This is a dog.-----
print : This is a dog.

input ? This is my dog.-----
print : This is my dog.

input ? My dog is in the box.-----
print : My dog is in the box.

input ? end-----
Ok

```

## 4 2 9 D H : 座標から V R A M のアドレスを求める.

アドレス	4 2 9 D H
機 能	キャラクタ座標から V R A M のアドレスを求める.
レジスタ	A, F, H, L

**解 説** レジスタ H にキャラクタ横座標 ( 1 ~ 8 0 ) を, レジスタ L にキャラクタ縦座標 ( 1 ~ 2 5 ) を与えてコールする事によって, C R T スクリーン上の任意のキャラクタ座標 ( H, L ) が実際には V R A M 上のどのアドレスに対応しているのかを計算して, レジスタ H L に格納してもどります.

8 0 / 4 0 桁モード, 2 5 / 2 0 行モード, カラー / 白黒モード等は全て考慮した上でアドレスが計算されます.

ただし, レジスタ H L に与える座標が適正範囲内でない場合には, “Illegal function call” ( F C Error ) エラーが発生します.

## サンプル

```

;
; --- display a box with character ---
;
;       ORG 0B900H
B900 2E05      LD  L,5
;
B902 2605      LOOPY: LD  H,5
;
B904 E5        LOOPX: PUSH HL
B905 CD9D42    CALL 429DH ; calculate address in video ram
B908 3623      LD  (HL),'#' ; send a character to video ram
B90A E1        POP  HL
;
B90B 24        INC  H
B90C 7C        LD  A,H
B90D FE15      CP   20+1
B90F 20F3      JR   NZ,LOOPX
;
B911 2C        INC  L
B912 7D        LD  A,L
B913 FE15      CP   20+1
B915 20EB      JR   NZ,LOOPY
;
B917 C9        RET
;

```

```
a=&hb900:call a
Ok
```

[illegible]

## 4 3 5 0 H : V R A M 転送および属性設定を行なう.

アドレス	4 3 5 0 H
------	-----------

機 能	V R A M 1 バイト転送および属性設定を行なう.
-----	-----------------------------

レジスタ	全て保存
------	------

解 説	レジスタ B のコードを, レジスタ H L が絶対アドレスで指定する V R A M 上の指定位置に転送すると同時に, アトリビュート・エリアのコントロールも行なってレジスタ C のアトリビュート・コードにより修飾するようにします.
-----	---

このサブルーチンを利用する事によって, N<sub>88</sub>-BASIC では, サポートされていないテキスト V R A M を用いたセミ・グラフィック機能等を活用する事もできます.

なお, レジスタ C に与えるアトリビュート・コードは, P C - 8 0 0 1 に使用されているものと同じ C R T コントローラ, N E C の  $\mu$  P D 3 3 0 1 に採用されているものです.

# サンプル

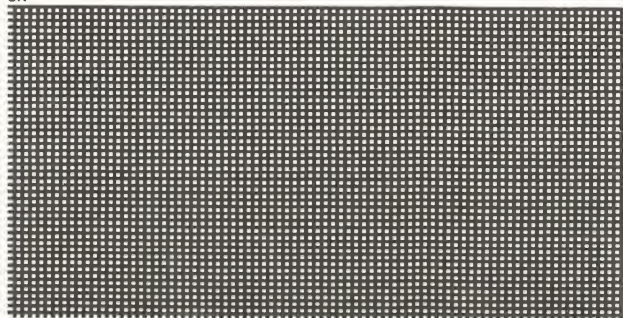
```

;
; --- use semi-graphic ---
;
; ORG 0B900H
;
B900 21B8F4 LD HL,0F3C8H+240
B903 0E17 LD C,23 ; number of line
B905 0650 LOOPY: LD B,80 ; number of column

B907 C5 LOOPX: PUSH BC
B908 E5 PUSH HL
;
B909 06F5 LD B,0F5H ; semi-graphic code
B90B 0E80 LD C,80H ; attribute for b/w mode
B90D CD5043 CALL 4350H ;*send a character with attribute
;
B910 E1 POP HL
B911 C1 POP BC
;
B912 23 INC HL ; next address in video-ram
B913 10F2 DJNZ LOOPX
;
B915 112800 LD DE,120-80
B918 19 ADD HL,DE ; skip attribute area 40 bytes
B919 0D DEC C
B91A 20E9 JR NZ,LOOPY
;
B91C C9 RET ; return to basic
;

```

console ,,,0:a=&hb900:call a  
Ok



## 4 4 5 2 H : V R A M上のキャラクタを調べる.

アドレス	4 4 5 2 H
機能	V R A M上のキャラクタを調べる.
レジスタ	A, F, B, C

**解 説** レジスタH Lの絶対アドレスで指定する. V R A M (ビデオ・ラム) に格納されているキャラクタ・コードをアキュムレータおよびレジスタBに, そのキャラクタを修飾しているアトリビュート・コードをレジスタCに, それぞれ格納してもどります.

アトリビュート・コードは, N E CのC R Tコントローラμ P D 3 3 0 1のもので, N-BASIC 等で利用しているものとコンパチブルです.

### サンプル

```

;
; --- erase not white characters ---
;
; ORG 0B900H
B900 2E01 ; LD L,1 ; initialize line counter
B902 2601 LOOPY: LD H,1 ; initialize column counter
;
B904 E5 LOOPX: PUSH HL
B905 CD9D42 CALL 429DH ; calculate address by coordinates
B908 CD5244 CALL 4452H ; *get a character with attribute
B90B 79 LD A,C ; attribute code in accumulator
B90C FEE8 CP 0E8H ; color mode, attribute for white
B90E 2802 JR Z,WHITE
;
B910 3620 LD (HL), ' ' ; send a space character to v-ram
;
B912 E1 WHITE: POP HL
B913 24 INC H
B914 7C LD A,H
B915 FE51 CP 80+1
B917 20EB JR NZ,LOOPX
;
B919 2C INC L
B91A 7D LD A,L
B918 FE1A CP 25+1
B91D 20E3 JR NZ,LOOPY
;
B91F C9 RET
;

```

```

console ,,,1:color 7:files 2
golf . 11 backup.n88 2 palet . 1 BASEBA. 7 xfiles.n88 1
majan *BIN 4 DStoSS.n88 2 yfiles.n88 1 majan . 11 DEMOUT.Y 6
DEMO . 3 cpaint. 1 quartz. 1 setinf.n88 1 format.n88 2

```

Ok

```

color 4:files 2
golf . 11 backup.n88 2 palet . 1 BASEBA. 7 xfiles.n88 1
majan *BIN 4 DStoSS.n88 2 yfiles.n88 1 majan . 11 DEMOUT.Y 6
DEMO . 3 cpaint. 1 quartz. 1 setinf.n88 1 format.n88 2

```

Ok

```

color 7:files 2
golf . 11 backup.n88 2 palet . 1 BASEBA. 7 xfiles.n88 1
majan *BIN 4 DStoSS.n88 2 yfiles.n88 1 majan . 11 DEMOUT.Y 6
DEMO . 3 cpaint. 1 quartz. 1 setinf.n88 1 format.n88 2

```

Ok

```

console ,,,1:color 7:files 2
golf . 11 backup.n88 2 palet . 1 BASEBA. 7 xfiles.n88 1
majan *BIN 4 DStoSS.n88 2 yfiles.n88 1 majan . 11 DEMOUT.Y 6
DEMO . 3 cpaint. 1 quartz. 1 setinf.n88 1 format.n88 2

```

Ok

```

color 4:files 2

```

```

golf . 11 backup.n88 2 palet . 1 BASEBA. 7 xfiles.n88 1
majan *BIN 4 DStoSS.n88 2 yfiles.n88 1 majan . 11 DEMOUT.Y 6
DEMO . 3 cpaint. 1 quartz. 1 setinf.n88 1 format.n88 2

```

Ok

```

a=&hb900:call a

```

Ok



## 4 4 7 2 H：スクリーン上のキャラクタを調べる。

**アドレス** 4 4 7 2 H

**機能** スクリーン上のキャラクタを調べる。

**レジスタ** A, F, B, C

**解説** レジスタHのキャラクタ横座標（1～80）とレジスタLのキャラクタ縦座標（1～25）で指定するCRTスクリーン上の座標（H, L）に表示されているキャラクタのコードをアキュムレータおよびレジスタBに、その座標を修飾しているアトリビュート・コードをレジスタCに、それぞれ格納してあります。

ただし、レジスタHおよびレジスタLに与える座標が不適当な場合には、“Illegal function call”（FC Error）エラーが発生します。

また、レジスタCに格納されるアトリビュート・コードは、PC-8801に使用されているCRTコントローラμPD3301のものです。

### サンプル

```

;
; --- change spaces into points on screen ---
;
;      ORG  0B900H
;
B900 2E01      LD  L,1          ; initialize line counter
;
B902 2601      LOOPY: LD  H,1      ; initialize column counter
;
B904 C07244    LOOPX: CALL 4472H    ; *get a character from video ram
B907 FE20      CP  ' '
B909 2007      JR  NZ,NOTSPC
;
B90B E5        PUSH HL
B90C C09D42    CALL 429DH          ; calculate address by coordinates
B90F 362E      LD  (HL),'.'        ; send a point to video ram
B911 E1        POP  HL
;
B912 24      NOTSPC: INC  H
B913 7C      LD  A,H
B914 FE51    CP  90+1
B916 20EC    JR  NZ,LOOPX
;
B918 2C      INC  L
B919 7D      LD  A,L
B91A FE1A    CP  25+1
B91C 20E4    JR  NZ,LOOPY
;
B91E C9      RET
;

```

a=&hb900:call.a.....  
Ok .....

## 4 4 7 D H : カーソル移動および1文字表示を行なう.

アドレス	4 4 7 D H
機能	カーソル移動および1文字表示を行なう.
レジスタ	A, F, B, C, H, L

**解 説** カーソルの横座標 (1 ~ 8 0) を E F 8 7 H 番地に, カーソルの縦座標 (1 ~ 2 5) を E F 8 6 H 番地に, 出力するコードをアキュムレータに, それぞれセットしてコールする事によって, C R T スクリーン上の指定位置に1バイトのキャラクタを表示する事ができます. アキュムレータのコードが 2 0 H 未満のコントロール・コードであれば, C R T スクリーンのクリアやカーソルの移動等コントロール・コードとしての動作を行ないます.



## 4 B 1 A H : ホット・スタートを行なう.

**アドレス** 4 B 1 A H

**機能** N<sub>88</sub>-BASICのホット・スタートを行なう.

**レジスタ**

**解説** N<sub>88</sub>-BASIC のホット・スタートを行ないます. CRTスクリーン等のイニシアライズは行ないませんが, N<sub>88</sub>-BASIC のプログラムは消去 (N E W) されてしまいます.

### サンプル

```

;
; --- clear text program or not ---
;
; ORG 0B900H
;
; START: LD HL,PROMPT
; CALL DSPMSG
; CALL 3583H ; input a character from keyboard
;
; RES 5,A ; convert into capital letter
; CP 'Y'
; JP NZ,NOTCLR
;
; LD HL,MSGYES ;
; CALL DSPMSG ; clear text and jump to basic
; JP 4B1AH ;
;
; NOTCLR:LD HL,MSGNOT ;
; CALL DSPMSG ; not clear and return to basic
; RET ;
;
; display message subroutine
;
; inputs : top address of message in register hl
; destroys : register a,f,h,l
;
; DSPMSG:LD A,(HL)
; AND A
; RET Z
; CALL 3E0DH ; display a character
; INC HL
; JR DSPMSG
;
; message area
;
;
; PROMPT:DEFB 'clear text program (y) ? ',0
;
; MSGYES:DEFB 'yes',0DH,0AH,'clear text program !!',0
;
; MSGNOT:DEFB 'no',0DH,0AH,'not clear text program !!',0
;

```

B966 636C6561  
B96A 72207465  
B96E 78742070  
B972 726F6772  
B976 616D2021  
B97A 2100

;

```
list
100 ,
110 , text program
120 ,
130 PRINT "text program here !!"
Ok
run
text program here !!
Ok
a=&hb900:call a
clear text program (y) ? no
not clear text program !!
Ok
list
100 ,
110 , text program
120 ,
130 PRINT "text program here !!"
Ok
run
text program here !!
Ok
a=&hb900:call a
clear text program (y) ? yes
clear text program !!
Ok
list
Ok
run
Ok
```

## 5 F C 8 H : ライン・インプットを行なう.

アドレス	5 F C 8 H
機 能	ライン・インプットを行なう.
レジスタ	A, F, B, C, D, E, H, L

**解 説** キーボードから1ライン入力し、N<sub>88</sub>-BASICのインプット・バッファ(E 9 B 9 H番地から)に入力データを入れてもどります。ライン・インプットは、キャリッジ・リターン・キーまたはストップ・キーが押された時点で入力終了となりますが、後者のストップ・キーによって入力を中断した場合にはキャリ・フラグ(CY)を1にセットしてもどります。

リターン時には、HLレジスタにE 9 B 8 Hが入っており、また入力データの末尾には0 0 Hを伴なっています。



# サンプル

```

;
; --- convert string into capitals ---
;
        ORG 0B900H
;
B900 112DB9 START: LD DE,PROMPT
B903 CD24B9 CALL DSPMSG
;
B906 CDC85F CALL 5FC8H ; line input subroutine
B909 D8 RET C
;
B90A 113BB9 LD DE,ANSWER
B90D CD24B9 CALL DSPMSG
;
B910 23 LOOP: INC HL
B911 7E LD A,(HL)
B912 A7 AND A
B913 28EB JR Z,START
;
B915 FE61 CP 'a' ;
B917 3806 JR C,DSPALY ;
B919 FE7B CP 'z'+1 ; check and convert into capitals
B91B 3002 JR NC,DSPALY ;
B91D CBAF RES 5,A ;
;
B91F CD0D3E DSPALY:CALL 3E0DH ; display a character
B922 18EC JR LOOP
;
; display message subroutine
;
; inputs : top address of message in register de
; destroys : register a,f,d,e
;
B924 1A DSPMSG:LD A,(DE)
B925 A7 AND A
B926 C8 RET Z
B927 CD0D3E CALL 3E0DH ; display a character
B92A 13 INC DE
B92B 18F7 JR DSPMSG
;
; message area
;
B92D 0D0A6120 PROMPT:DEFB 0DH,0AH,'a string ? ',0
B931 73747269
B935 6E67203F
B939 2000
B93B 63617069 ANSWER:DEFB 'capitals : ',0
B93F 74616C73
B943 203A2000
;

```

a=&hb900:call a

```

a string ? This is a dog.
capitals : THIS IS A DOG.
a string ? This is my dog.
capitals : THIS IS MY DOG.
a string ? My dog is in the box.
capitals : MY DOG IS IN THE BOX.
a string ?
capitals :
a string ?
Ok

```

## 5 F 9 2 H : スクリーン・エディトを行なう.

アドレス	5 F 9 2 H
機 能	スクリーン・エディトを行なう.
レジスタ	A, F, B, C, D, E, H, L

**解 説** スクリーン・エディト方式によってキーボードから1ラインのデータを入力して、N88-BASICのインプット・バッファ(E 9 B 9 H番地から)に格納してもどります。ただし入力データは254バイトまで有効で、254バイトを越えて入力した場合には先頭からの254バイトが入力データと見なされません。

データの入力は、キャリッジ・リターン・キー(CTRL-M)またはストップ・キー(CTRL-C)の入力によって終了しますが、前者の場合にはキャリ・フラグ(CY)を0にリセットし、後者の入力によって中断した場合にはキャリ・フラグ(CY)を1にセットしてもどります。

このサブルーチンからのリターン時には、レジスタHLにE 9 B 8 H(インプット・バッファのアドレス-1)が、アキュムレータには最後に入力されたキャラクタのコード(0 3 Hまたは0 D H)が与えられます。

### サンプル

```

;
; --- micro word processor ---
;
;      ORG      0B900H
;
B900 21C2B9      LD      HL,TITLE
B903 1806        JR      DISPLAY
;
B905 3E07      ERROR: LD      A,07H
B907 DF        RST      18H
;
B908 21D9B9      START0:LD    HL,PROMPT
B90B CDB8B9      DISPLAY:CALL DSPMSG
;
B90E CD925F      START: CALL  5F92H          ;*screen editor
B911 38FB        JR      C,START
B913 23          INC     HL
B914 7E          LD      A,(HL)
B915 CBAF        RES     5,A          ; convert into capital letter
B917 FE41        CP      'A'
B919 38EA        JR      C,ERROR
B91B FE58        CP      'Z'+1
B91D 30E6        JR      NC,ERROR
;
B91F 21E2B9      LD      HL,TABLE          ; top address of command table
B922 0604        LD      B,4              ; set number of command
B924 11B9E9      START1:LD    DE,0E9B9H    ; input buffer of n88-basic
B927 1A          START2:LD    A,(DE)
B928 CBAF        RES     5,A          ; convert into capital letter
B92A BE          CP      (HL)
B92B 2007        JR      NZ,START3
B92D A7          AND     A

```

```

B92E 2828      JR  Z,YESCOM
B930 23        INC  HL
B931 13        INC  DE
B932 18F3      JR  START2

;
B934 34        START3: INC  (HL)
B935 35        DEC  (HL)
B936 2803      JR  Z,START4
B938 23        INC  HL
B939 18F9      JR  START3

;
B93B 23        START4: INC  HL
B93C 23        INC  HL
B93D 23        INC  HL
B93E 10E4      DJNZ START1

;
B940 11B9E9    EDIT:  LD  DE,0E9B9H      ; input buffer of n88-basic
B943 1A        LD  A,(DE)
B944 13        INC  DE
B945 08        EX  AF,AF'
B946 1A        LD  A,(DE)
B947 13        INC  DE
B948 FE3A      CP  ':'
B94A 20B9      JR  NZ,ERROR
B94C 08        EX  AF,AF'
B94D CDB1B9    CALL CALADR      ; calculate top address of buffer
B950 EB        EX  DE,HL
B951 01FF00    LD  BC,0FFH
B954 EDB0      LDIR
B956 18B6      JR  START

;
B958 23        YESCOM: INC  HL
B959 5E        LD  E,(HL)
B95A 23        INC  HL
B95B 56        LD  D,(HL)
B95C EB        EX  DE,HL
B95D E9        JP  (HL)

;
; new command section
;
B95E 2100BA    NEW:   LD  HL,BUFFER
B961 AF        XOR  A
B962 061A      LD  B,'z'-'a'+1
B964 77        NEW1: LD  (HL),A
B965 24        INC  H
B966 10FC      DJNZ NEW1
B968 189E      JR  START0

;
; print command section
;
B96A 3E01      PRINT: LD  A,1
B96C 324CE6    LD  (0E64CH),A      ; select printer for output

;
B96F 3E61      LD  A,'a'
B971 CDA2B9    PRINT1:CALL PRNLIN      ; set first line number in acc.
;                                     ; print out a line to printer
;
B974 47        LD  B,A
B975 CDC235    CALL 35C2H      ; check stop key
B978 388E      JR  C,START0      ; stop key, return to main routine
B97A 78        LD  A,B

;
B97B 3C        INC  A
B97C FE7B      CP  'z'+1
B97E 20F1      JR  NZ,PRINT1      ; increment line number in acc.
;                                     ; check line number end

;
B980 AF        XOR  A
B981 324CE6    LD  (0E64CH),A      ; select crt for output
B984 1882      JR  START0

;
; list command section
;
B986 3E61      LIST:  LD  A,'a'
B988 CD9CB9    LIST1: CALL LSTLIN      ; set first line number
;                                     ; display a line
;

```

```

B98B 47          LD  B,A
B98C C0C235      CALL 35C2H          ; check stop key
B98F DA08B9      JP  C,START0        ; stop key, return to main routine
B992 78          LD  A,B

;
B993 3C          INC  A              ; increment line number in acc.
B994 FE7B        CP   'z'+1          ; check line number end
B996 20F0        JR   NZ,LIST1
B998 C308B9      JP   START0

;
; system command section
;
B99B C9          SYSTEM:RET          ; return to basic
;
; list one line with line number subroutine
;
; inputs : line number from a to z in accumulator
; destroys : register h,l
;
B99C DF          LSTLIN:RST 18H        ; output line number
B99D F5          PUSH AF
B99E 3E3A        LD   A,';'
B9A0 DF          RST 18H
B9A1 F1          POP  AF

;
; list one line without line number subroutine
;
; inputs : line number from a to z in accumulator
; destroys : register h,l
;
B9A2 F5          PRNLIN:PUSH AF
B9A3 CDB1B9      CALL CALADR          ; calculate top address of buffer
B9A6 CDBBB9      CALL DSPMSG

;
B9A9 21DFB9      LD   HL,CRLF
B9AC CDBBB9      CALL DSPMSG

;
B9AF F1          POP  AF
B9B0 C9          RET

;
; calculate top address of buffer subroutine
;
; inputs : line number from a to z in accumulator
; outputs : top address of buffer in register hl
; destroys : register a,f,h,l
;
B9B1 CBAF        CALADR:RES 5,A
B9B3 D641        SUB  'A'
B9B5 2100BA      LD   HL,BUFFER
B9B8 84          ADD  A,H
B9B9 67          LD   H,A
B9BA C9          RET

;
; display message subroutine
;
; inputs : top address of message in register hl
; destroys : register a,f,h,l
;
B9BB 7E          DSPMSG:LD  A,(HL)
B9BC A7          AND  A
B9BD C8          RET  Z
B9BE DF          RST 18H              ; output a character
B9BF 23          INC  HL
B9C0 18F9        JR   DSPMSG

;
; message area
;
B9C2 776F7264    TITLE: DEFB 'word processing system '
B9C6 2070726F
B9CA 63657373
B9CE 696E6720
B9D2 73797374
B9D6 656D20
B9D9 68657265    PROMPT:DEFB 'here.',0FFH

```

```

B9DD 2EFF
B9DF 0D0A00 CRLF: DEFB 0DH,0AH,0
;
; command table
;
B9E2 4C495354 TABLE: DEFB 'LIST',0
B9E6 00
B9E7 86B9 DEFW LIST
B9E9 5052494E DEFB 'PRINT',0
B9ED 5400
B9EF 6AB9 DEFW PRINT
B9F1 4E455700 DEFB 'NEW',0
B9F5 5EB9 DEFW NEW
B9F7 53595354 DEFB 'SYSTEM',0
B9FB 454D00
B9FE 98B9 DEFW SYSTEM
;
; buffer for edit
;
BA00 BUFFER:EQU 0BA00H
;

```

# << MICRO WORD PROCESSOR SPECIFICATIONS >>

```

program area : from b900h to b9ffh
data area : from ba00h to d3ffh

operation : screen edit
line number : from a to z, or from A to Z

title message : micro processing system here.
prompt message : here.
error message : ring the bell

```

## << COMMAND >>

```

list : display text
print : output text to printer
new : clear text
system : exit from word processing system

```

## << ATTENTION >>

this specifications printed with this program.

## 6 F 6 B H : C R Tスクリーンのモード設定を行なう.

アドレス	6 F 6 B H
機 能	C R Tスクリーンのモード設定を行なう.
レジスタ	A, F, B, C, D, E

**解 説** 以下に示す入力パラメータにより, C R Tスクリーンの設定を行なってもどるサブルーチンで, テキスト画面に関するほとんど全ての設定を1度に行なう事ができます.

レジスタBにはスクリーンの桁数/ラインを与えます.

レジスタCにはスクリーンのライン数/画面を与えます.

E 6 B 2 H番地にはスクロール開始行 (1 ~ 2 5) を与えます.

E 6 B 3 H番地にはスクロール終了行 (1 ~ 2 5) を与えます.

E 6 B 9 H番地にはカラー・モードか否か (F F H / 0 0 H) を与えます.

N<sub>88</sub> - B A S I Cでは "Illegal function call" (F C Error) エラーが発生する様な (2 0 桁×1 0 行等) 設定も可能ですが, スクロールする範囲が最低1行は存在しなくてはけません.

また, カラー・モード/白黒モードのモード切り換えが行なわれても, ファンクション・コード (カラー・コード) までは変化しないため, テキスト画面に異常なパターンが表示されたり, また何も表示されなかったりする場合も有り得ます. その様な場合にはE 6 B 4 H番地等に適当なアトリビュート・コードを与えるか, N<sub>88</sub> - B A S I CのC O L O Rによってファンクション・コードを変更してください.

# サンプル

```

;
; --- change crt screen mode ---
;
        ORG 0B900H
;
B900 216FB9      LD HL,WIDTH
B903 CD66B9      CALL DSPMSG
B906 CD43B9      CALL INDEC
B909 43          LD B,E          ; set width x in register b
;
B90A CD61B9      CALL DSPCOM
B90D CD43B9      CALL INDEC
B910 4B          LD C,E          ; set width y in register c
;
B911 2178B9      LD HL,CONSOL
B914 CD66B9      CALL DSPMSG
B917 CD43B9      CALL INDEC
B91A 1C          INC E
B91B 7B          LD A,E
B91C 32B2E6      LD (0E6B2H),A    ; set scroll start line (1-25)
;
B91F CD61B9      CALL DSPCOM
B922 CD43B9      CALL INDEC
B925 3AB2E6      LD A,(0E6B2H)
B928 83          ADD A,E
B929 3D          DEC A
B92A 32B3E6      LD (0E6B3H),A    ; set scroll end line (1-25)
;
B92D CD61B9      CALL DSPCOM
B930 CD61B9      CALL DSPCOM
B933 CD43B9      CALL INDEC
B936 AF          XOR A
B937 1C          INC E
B938 1D          DEC E
B939 2802        JR Z,BW
B93B 3EFF        LD A,0FFH
B93D 32B9E6      BW: LD (0E6B9H),A ; set color mode or b/w mode
;
B940 C36B6F      JP 6F6BH          ;*change crt mode and return
;
; input a decimal number subroutine
;
; outputs : binary code less than 256 in register e
; destroys : register a,f,e
;
B943 1E00      INDEC: LD E,0          ; initialize a input number
B945 3E30      LD A,'0'          ; initialize a input character
;
B947 F5      INDEC1:PUSH AF          ;
B948 7B      LD A,E          ;
B949 87      ADD A,A          ;
B94A 87      ADD A,A          ;
B94B 83      ADD A,E          ;
B94C 87      ADD A,A          ; e <-- e * 10 + input number
B94D 5F      LD E,A          ;
B94E F1      POP AF          ;
B94F D630      SUB '0'          ;
B951 83      ADD A,E          ;
B952 5F      LD E,A          ;
;
B953 CD8335      CALL 3583H          ; input a character from keyboard
;
B956 FE3A      CP '9'+1
B958 D0      RET NC
B959 FE30      CP '0'
B95B D8      RET C
;
B95C CD0D3E      CALL 3E0DH          ; echo-back a character
B95F 18E6      JR INDEC1

```





## 7 1 D 9 H：タイマヘデータの書き込みを行なう。

アドレス	7 1 D 9 H
機能	タイマヘデータの書き込みを行なう。
レジスタ	A, F, B, C, D, E, H, L

**解 説** ワーク・エリア (F 0 0 D H ~ F 0 1 2 H 番地) のデータを、ハード・タイマにセットしてもどります。それぞれのデータは B C D コードとしてワーク・エリアに与えなければなりません。MONTH (F 0 1 1 H 番地) のみは例外で、通常のバイナリ・コード (0 1 H ~ 0 C H) で与えます。

以下に、タイマ制御用のワーク・エリアを示します。

F 0 0 D H 番地	←	SECOND (秒: B C D コード)
F 0 0 E H 番地	←	MINUTE (分: B C D コード)
F 0 0 F H 番地	←	HOURL (時: B C D コード)
F 0 1 0 H 番地	←	DAY (日: B C D コード)
F 0 1 1 H 番地	←	MONTH (月: バイナリ・コード)
F 0 1 2 H 番地	←	YEAR (年: B C D コード)

# サンプル

```

;
; --- write to timer ---
;
;       ORG 0B900H
;
; input date section
;
B900 216EB9      LD HL,PDATE
B903 CD65B9      CALL DSPMSG
B906 CD925F      CALL 5F92H      ; screen editor
B909 D8          RET C          ; break code, return to basic
;
B90A 1112F0      LD DE,0F012H   ; end address of date buffer
B90D 0603      LD B,3
B90F C5      LDATE: PUSH BC
B910 23      INC HL
B911 46      LD B,(HL)
B912 23      INC HL
B913 4E      LD C,(HL)
;
B914 F1      POP AF
B915 F5      PUSH AF
B916 3D      DEC A
B917 3D      DEC A      ; check minute or other
B918 2805     JR Z,MONTH
;
B91A CD46B9      CALL CHRBCD
B91D 1803      JR SKIP          ; not month, convert into bcd code
;
B91F CD57B9      MONTH: CALL CHRBIN
B922 12      SKIP: LD (DE),A      ; month, convert into binary code
B923 18      DEC DE
B924 C1      POP BC
B925 10E8      DJNZ LDATE
;
; input time section
;
B927 217FB9      LD HL,PTIME
B92A CD65B9      CALL DSPMSG
B92D CD925F      CALL 5F92H      ; screen editor
B930 D8          RET C          ; break code, return to basic
;
B931 110FF0      LD DE,0F012H-3 ; end address of date buffer
B934 0603      LD B,3
B936 C5      LTIME: PUSH BC
B937 23      INC HL
B938 46      LD B,(HL)
B939 23      INC HL
B93A 4E      LD C,(HL)
;
B93B CD46B9      CALL CHRBCD      ; convert into bcd code
;
B93E 12      LD (DE),A
B93F 18      DEC DE
B940 C1      POP BC
B941 10F3      DJNZ LTIME
;
; write to timer section
;
B943 C3D971      JP 71D9H      ;*write to timer
;
; convert character code into bcd code subroutine
;
;       inputs : two characters in register bc
;       outputs : bcd code in accumulator
;       destroys : register a,f,b
;
B946 78      CHRBCD:LD A,B
B947 D630      SUB '0'
B949 47      LD B,A
B94A CB20      SLA B

```

```

B94C CB20      SLA  B
B94E CB20      SLA  B
B950 CB20      SLA  B
B952 79        LD   A,C
B953 D630      SUB  '0'
B955 80        ADD  A,B
B956 C9        RET

;
; convert character code into binary code subroutine
;
; inputs : two characters in register bc
; outputs : binary code in accumulator
; destroys : register a,f,b
;
B957 78        CHRBIN:LD  A,B
B958 D630      SUB  '0'

;
B95A 47        LD   B,A
B95B 87        ADD  A,A
B95C 87        ADD  A,A
B95D 80        ADD  A,B
B95E 87        ADD  A,A

;
B95F 47        LD   B,A
B960 79        LD   A,C
B961 D630      SUB  '0'
B963 80        ADD  A,B
B964 C9        RET

;
; display message subroutine
;
; destroys : register a,f,h,l
;
B965 7E        DSPMSG:LD  A,(HL)
B966 A7        AND  A
B967 C8        RET  Z
B968 CD0D3E    CALL 3E0DH
B96B 23        INC  HL
B96C 18F7      JR   DSPMSG

;
; message area
;
B96E 64617465  PDATE: DEFB 'date - yymmdd ? ',0
B972 202D2079
B976 796D6D64
B97A 64203F20
B97E 00
B97F 74696D65  PTIME: DEFB 'time - hhmmss ? ',0
B983 202D2068
B987 686D6D73
B98B 73203F20
B98F 00

;

print date$:print time$
82/09/01
19:05:06
Ok
a=&hb900:call a
date - yymmdd ? 011223
time - hhmmss ? 122334
Ok
print date$:print time$
01/12/23
12:23:39
Ok

```

## 7 2 C D H : I / O を N<sub>88</sub>-BASIC 用に設定する.

アドレス	7 2 C D H
機能	I / O を N <sub>88</sub> -BASIC 用に設定する.
レジスタ	A, F, B, C, D, E, H, L

**解 説** P C - 8 8 0 1 の各 I / O ポートを, N<sub>88</sub>-BASIC 用にイニシアライズしてもどります.

### サンプル

```

;
; --- initialize i/o for n88-basic ---
;
;      ORG 0B900H
;
B900 2124B9      LD HL,PROMPT
B903 CD1BB9      CALL DSPMSG
B906 CD8335      CALL 3583H          ; input a character from keyboard
B909 CD0D3E      CALL 3E0DH          ; echo-back a character
B90C CBAF        RES 5,A            ; convert into capital letter
B90E FE59        CP 'Y'
B910 C0          RET NZ              ; return to basic
B911 CDC072      CALL 72CDH          ;*initialize i/o for n88-basic
B914 2148B9      LD HL,MESSAGE
B917 CD1BB9      CALL DSPMSG
B91A C9          RET                ; return to basic

;
; display message subroutine
;
;      inputs : top address of message in register hl
;      destroys : register a,f,h,l
;
B91B 7E          DSPMSG:LD A,(HL)
B91C A7          AND A
B91D C8          RET Z
B91E CD0D3E      CALL 3E0DH          ; display a character
B921 23          INC HL
B922 18F7        JR DSPMSG

;
; message area
;
B924 696E6974    PROMPT:DEFB 'initialize i/o'
B928 69616C69
B92C 7A652069
B930 2F6F
B932 20666F72    DEFB ' for n88-basic (y) ? ',0
B936 206E3838
B93A 2D626173
B93E 69632028
B942 7929203F
B946 2000
B948 696E6974    MESSAGE:DEFB 'initialized !!',0
B94C 69616C69
B950 7A656420
B954 212100
;

```

```

a=&hb900:call a
initialize i/o for n88-basic (y) ? y
initialized !!

```

## 780FH: ホット・スタートを行なう(2).

**アドレス** 780FH

**機能** N88-BASICのホット・スタートを行なう(2).

**レジスタ**

**解説** N88-BASICのホット・スタート・アドレスです。ホット・スタートを行なってもBASICのテキストは保存されており変数の値等も変化しませんが、ペリフェラルのイニシアライズを行なうためCRTスクリーンはテキスト、グラフィック共に消去されます。

### サンプル

```

;
; --- hot start or return ---
;
        ORG 0B900H
;
B900 212EB9  START: LD  HL,PROMPT
B903 CD25B9  CALL DSPMSG
;
B906 CD8335  LOOP:  CALL 3583H          ; input a character from keyboard
B909 CBAF    RES 5,A                ; convert into capital letter
B90B FE48    CP 'H'
B90D 2806    JR Z,HOT
B90F FE52    CP 'R'
B911 280B    JR Z,RETURN
B913 18F1    JR LOOP
;
B915 2148B9  HOT:   LD  HL,MSGHOT      ;
B918 CD25B9  CALL DSPMSG              ; jump to hot start
B91B C30F78  JP 780FH                 ;
;
B91E 2158B9  RETURN:LD HL,MSGRET      ;
B921 CD25B9  CALL DSPMSG              ; return to basic
B924 C9      RET                      ;
;
; display message subroutine
;
; inputs : top address of message in register hl
; destroys : register a,f,h,l
;
B925 7E      DSPMSG:LD  A,(HL)
B926 A7      AND  A
B927 C8      RET  Z
B928 CD0D3E  CALL 3E0DH              ; display a character
B92B 23      INC  HL
B92C 18F7    JR  DSPMSG
;
; message area
;
B92E 686F7420 PROMPT:DEFB 'hot start or return (h/r) ? ',0
B932 73746172
B936 74206F72
B93A 20726574
B93E 75726E20
B942 28682F72
B946 29203F20
B94A 00
B94B 686F7420 MSGHOT:DEFB 'hot start !!!',0
B94F 73746172
B953 74202121

```

```

B957 00
B958 72657475 MSGRET:DEFB 'return to basic !!',0
B95C 726E2074
B960 6F206261
B964 73696320
B968 212100
      ;

```

```

list
100 /
110 / text program
120 /
130 PRINT "text program here !!"
Ok
run
text program here !!
Ok
a=&hb900:call a
hot start or return (h/r) ? return to basic !!
Ok
list
100 /
110 / text program
120 /
130 PRINT "text program here !!"
Ok
run
text program here !!
Ok
a=&hb900:call a
hot start or return (h/r) ? hot start !!
Ok
list
100 /
110 / text program
120 /
130 PRINT "text program here !!"
Ok
run
text program here !!
Ok

```



## 7ED0H：CMTを入力用にイニシアライズする。

アドレス	7ED0H
------	-------

機能	CMTインタフェースを入力用にイニシアライズする。
----	---------------------------

レジスタ	A, F, H, L
------	------------

解説	転送速度1200ボーを利用する場合にはF009H番地にFBHを、600ボーを利用する場合にはFAHを与えてコールすると、USART8251をCMTの入力用にセットし、内蔵モータ・リレーをON(MOTOR1)後キャリア信号を検出してもどります。
----	---

## 7F15H：CMTからの入力をクローズする。

アドレス	7F15H
------	-------

機能	CMTインタフェースからの入力をクローズする。
----	-------------------------

レジスタ	A, F
------	------

解説	CMTインタフェースからの入力をクローズし、内蔵モータ・リレーをOFF(MOTOR0)にしてもどります。
----	--

## 7F1AH：CMTへの出力をクローズする。

アドレス	7F1AH
------	-------

機能	CMTインタフェースへの出力をクローズする。
----	------------------------

レジスタ	A, F
------	------

解説	CMTインタフェースへの出力をクローズしてしばらくの間時間待ちを行ない、内蔵モータ・リレーをOFF(MOTOR0)にしてもどります。
----	--

## 7 F 3 5 H : 内蔵リレーのコントロールを行なう.

アドレス 7 F 3 5 H

機能 内蔵リレーのコントロールを行なう.

レジスタ A, F

**解 説** アキュムレータのデータが0の場合には内蔵リレーをOFFに (MOTOR 0), アキュムレータのデータが0以外の場合には内蔵リレーをONに (MOTOR 1) します.

PC-8801では, 出力ポート30H番地の第3ビットを直接コントロールする事によって, 内蔵リレーのコントロールを行なう事ができますが他のビットを変化させないように注意する事がが必要です. 出力ポート30H番地に出力したデータは, N88-BASICではE6C0H番地に保存されています.

### サンプル

```

;
; --- control motor relay ---
;
; ORG 0B900H
;
B900 213EB9 START: LD HL,PROMPT
B903 CD2AB9 CALL DSPMSG
B906 CD8335 CALL 3583H ; input a character from keyboard
B909 FE03 CP 03H ; check break code,
B90B C8 RET Z ; return to basic
;
B90C FE30 CP '0'
B90E 200B JR NZ,NOTOFF
;
B910 2159B9 LD HL,MSG0 ;
B913 CD2AB9 CALL DSPMSG ; motor relay off
B916 CD33B9 CALL MOTOR0 ;
B919 '18E5 JR START
;
B91B FE31 NOTOFF:CP '1'
B91D 20E1 JR NZ,START
;
B91F 216CB9 LD HL,MSG1 ;
B922 CD2AB9 CALL DSPMSG ; motor relay on
B925 CD33B9 CALL MOTOR1 ;
B928 18D6 JR START
;
; display message subroutine
;
B92A 7E DSPMSG:LD A,(HL)
B92B A7 AND A
B92C C8 RET Z
B92D CD0D3E CALL 3E0DH ; display a character
B930 23 INC HL
B931 18F7 JR DSPMSG
;
;
; motor relay off subroutine
;
B933 AF MOTOR0:XOR A
B934 CD357F CALL 7F35H ;*control motor relay
B937 C9 RET

```

```

;
; motor relay on subroutine
;
B938 3E01    MOTOR1:LD    A,1
B93A CD357F    CALL 7F35H    ;*control motor relay
B93D C9      RET

;
; message area
;
B93E 0D0A6869 PROMPT:DEFB 0DH,0AH,'hit key, motor 0 or 1 ? ',0
B942 74206B65
B946 792C206D
B94A 6F746F72
B94E 2030206F
B952 72203120
B956 3F2000
B959 6D6F746F MSG0: DEFB 'motor relay off !!',0
B95D 72207265
B961 6C617920
B965 6F666620
B969 212100
B96C 6D6F746F MSG1: DEFB 'motor relay on !!',0
B970 72207265
B974 6C617920
B978 6F6E2021
B97C 2100

```

a=&hb900:call a

```

hit key, motor 0 or 1 ?
hit key, motor 0 or 1 ?
hit key, motor 0 or 1 ? motor relay off !!
hit key, motor 0 or 1 ? motor relay on !!
hit key, motor 0 or 1 ? motor relay on !!
hit key, motor 0 or 1 ? motor relay off !!
hit key, motor 0 or 1 ?
hit key, motor 0 or 1 ? motor relay off !!
hit key, motor 0 or 1 ? motor relay on !!
hit key, motor 0 or 1 ?
Ok

```

## 7F4DH：CMTを出力用にイニシアライズする。

アドレス	7F4DH
------	-------

機能	CMTインタフェースを出力用にイニシアライズする。
----	---------------------------

レジスタ	A, F, B
------	---------

解説	転送速度1200ボーを利用する場合にはF009H番地にFBHを、600ボーを利用する場合にはFAHを、それぞれ与えてコールすると、USART8251をCMTの入力用にセットし、内蔵モータ・リレーをOFF(MOTOR 0)後、しばらくの間時間待ちを行なってもどります。
----	---

## 7F87H：CMTからの1バイト入力を行なう。

アドレス	7F87H
------	-------

機能	CMTインタフェースからの1バイト入力を行なう。
----	--------------------------

レジスタ	A, F
------	------

解説	CMTインタフェースから1バイトのデータを入力し、アキュムレータに格納してもどります。ストップ・キーの入力があればCMTインタフェースからのデータ入力を中断します。
----	--

## 7FD0H：CMTへの1バイト出力を行なう。

アドレス	7FD0H
------	-------

機能	CMTインタフェースへの1バイト出力を行なう。
----	-------------------------

レジスタ	A, F
------	------

解説	CMTインタフェースへ、アキュムレータのデータ1バイトを出力してもどります。ストップ・キーの入力があれば、CMTインタフェースへのデータ出力を中断します。
----	---

# ***I/O PORT MAPPING***



本章では、PC-8801が使用している各I/Oポートの意味を示します。  
各ポートはできる限りポート・アドレスによってソーティングし、分類しました。

<b>ポ ー ト</b>	各I/Oポートのポート・アドレスを示します。
<b>I / O</b>	各I/OポートがINPUT（入力）ポートであるかOUTPUT（出力）ポートであるかの区別を示します。
<b>機 能</b>	各I/Oポートの使用目的を簡単に示します。
<b>解 説</b>	各I/Oポートの詳しい使用方法、各ビットの意味を説明した上で、場合によっては関連した予備知識も示します。

なお、I/Oポート・アドレスはPC-8001の場合と同じく第7ビットのシステム・コードでPC-8801内部のI/Oインタフェースを用いるか否かを区別し、第6ビット～第4ビットのデバイス・ナンバによって具体的にI/Oインタフェースの種類を選択します。下位4ビットのオーダ・コードは選択したインタフェースに関して実際の働きを指定するものです。

ポート・アドレス 上位 下位	I/Oインタフェース
0 0 0 0	キーボード
0 0 0 1	プリンタ
0 0 1 0	USART 8 2 5 1
0 0 1 1	システム・コントロール
0 1 0 0	システム・コントロール
0 1 0 1	CRTコントローラ
0 1 1 0	DMAコントローラ
0 1 1 1	メモリ・バンク
1 0 0 0	拡張バス・スロット
1 0 0 1	拡張バス・スロット
1 0 1 0	拡張バス・スロット
1 0 1 1	拡張バス・スロット
1 1 0 0	拡張バス・スロット
1 1 0 1	拡張バス・スロット
1 1 1 0	インタラプト
1 1 1 1	フロッピ・ディスク



## 00H:キー・マトリクス of データを入力する。

ポ ー ト 00H~0BH

I / O INPUT

機 能 キー・マトリクス of データを入力する。

**解 説** PC-8801 のキーボードは、エンコード等が用いられておらず、一般的なソフトウェア・キー・スキャンによって押されたキーの判別を行なっています。N88-BASICではこれらの処理をインタラプトで行なっているため、バッファの許す範囲(31文字)で先行入力を行なう事が可能です。

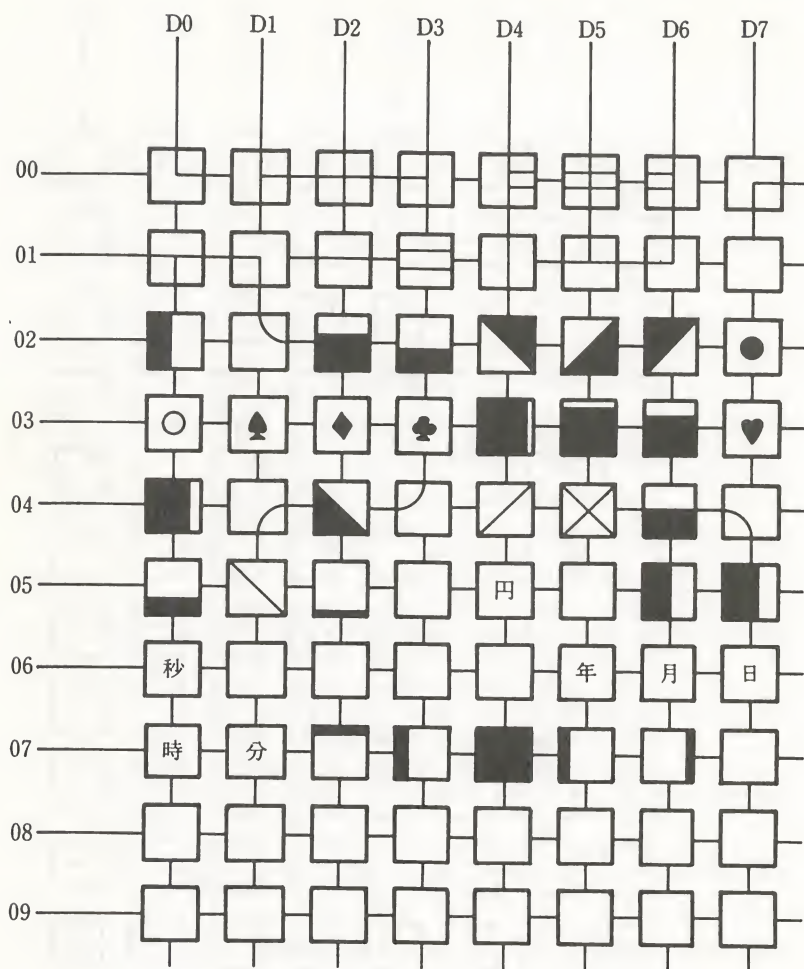
キーボード・マトリクス of 構成は、「PC-8801 USER'S MANUAL」等に掲載されていますが、次のプログラムを走らせる事によって押されたキーと入力ポートデータの関係を調べることができます。

### サンプル

```
100 '  
110 ' check key scanning port  
120 '  
130 DIM PORT(&HB)  
140 ON STOP GOSUB *BREAK:STOP ON  
150 WIDTH 40,25:CONSOLE 0,25:PRINT CHR$(12);  
160 PRINT 'port 00 01 02 03 04 05 06 07 08 09 0a 0b';  
170 PRINT '-----';  
180 *LOOP  
190 FLAG=0  
200 FOR I=&H0 TO &HB  
210   PORT(I)=INP(I)  
220   IF PORT(I)<>&HFF THEN BEEP 1:FLAG=-1:BEEP 0  
230 NEXT  
240 IF NOT FLAG THEN *LOOP  
250 '  
260 PRINT TAB(5);  
270 FOR I=&H0 TO &HB  
280   PRINT RIGHT$('0'+HEX$(PORT(I)),2);' ';  
290 NEXT  
300 GOTO *LOOP  
310 '  
320 *BREAK  
330 STOP OFF  
340 RETURN
```

```
run  
port 00 01 02 03 04 05 06 07 08 09 0a 0b  
-----  
FF FF FD FF FF FF FF FF FF FF FF FF  
FF FF FB FF FF FF FF FF FF FF FF FF  
FF FF FF FF FF FF FF FF FE FF FF  
^C  
Break in 200  
Ok
```

		(データ・バス)							
		D0	D1	D2	D3	D4	D5	D6	D7
(アドレス・バス)									
00		0	1	2	3	4	5	6	7
01		8	9	*	+	=	,	.	RETURN
02		@	A チ	B コ	C ソ	D シ	E イ	F ハ	G キ
03		H ク	I ニ	J マ	K ノ	L リ	M モ	N ミ	O ラ
04		P セ	Q タ	R ス	S ト	T カ	U ナ	V ヒ	W テ
05		X サ	Y ン	Z ツ	[。「	¥	]ム」	^へ	_ = ホ
06		0 フ	1 / ヌ	2 フ	3 # ア	4 \$ ウ	5 % エ	6 & オ	7 ▼ ヤ
07		8 ( ユ	9 ) ヨ	: * ケ	; + レ	< , ネ	> . ル	? / メ	ー ロ
08		HOME CLR	↑	→	INS DEL	GRAPH	カナ	SHIFT	CTRL
09		STOP	f・1	f・2	f・3	f・4	f・5	SPACE	ESC
0A		HTAB	↓	←	HELP	COPY	—	/	CAPS LOCK
0B		ROLL UP	ROLL DOWN						



## 10H：プリンタおよびタイマヘデータを出力する。

ポ ー ト	10H
I / O	OUTPUT
機 能	プリンタおよびタイマヘデータを出力する。

**解 説** プリンタヘデータを出力するためのシステム・サブルーチンはN-BASIC、N88-BASIC共に用意されています。

N88-BASICでは、アキュムレータのデータをプリンタへ出力する3ED4H番地のシステム・サブルーチンを使う事もできますし、リスタートでコールする事のできる0018H番地の出力サブルーチンもE64CH番地のフラグをセットする事でプリンタに出力します。

しかし、直接入出力ポートをコントロールしてプリンタとのハンドシェイクを行なうためには、入力ポート40H番地の第0ビットにBUSY信号が入力しており、出力ポート40H番地の第0ビットで $\overline{\text{PSTB}}$ （プリンタ・ストロブ）信号をコントロールする事ができます。データの出力はもちろん、この10H番地の出力ポートを利用して行ないます。

また、下位4ビットはタイマのコントロールにも使用されます。その場合には下位3ビットがコマンド出力用に、第3ビットがシリアル・データの出力用になります。タイマをコントロールするためには、ほかに出力ポート40H番地の第2ビット（CLK）および第1ビット（CSTB）の制御が必要です。

# サンプル

```

100 /
110 / print out characters
120 /
130 A=&H1B:GOSUB *PRNCHR:A=ASC('B'):GOSUB *PRNCHR
140 FOR I=1 TO 17
150   A=ASC(MID$(" 0123456789abcdef",I,1)):GOSUB *PRNCHR
160 NEXT I
170 GOSUB *PRNCRLF
180 /
190 FOR Y=0 TO &HF
200   A=ASC(MID$("0123456789abcdef",Y+1,1)):GOSUB *PRNCHR
210   FOR X=0 TO &HF
220     A=X*&H10+Y:IF A<&H20 THEN A=&H20
230     GOSUB *PRNCHR
240   NEXT X
250   GOSUB *PRNCRLF
260 NEXT Y
270 A=&H1B:GOSUB *PRNCHR:A=ASC('A'):GOSUB *PRNCHR
280 END
290 /
300 *PRNCRLF
310 A=&HD:GOSUB *PRNCHR
320 A=&HA
330 /
340 *PRNCHR
350 IF (INP(&H40) AND &H1)<>0 THEN *PRNCHR
360 OUT &H10,A
370 A=PEEK(&HE&C1)
380 A=A AND &HFE:OUT &H40,A
390 A=A OR &H1:OUT &H40,A
400 RETURN

```

```

run
0123456789abcdef
0 0@P`p_+ -タミ=X
1 !1AQaQ~ro7チムH門
2 "2BRbr~!「イツメキ
3 #3CScs~トクテモイ月
4 $4DTdt~、イトイ目。
5 %5EUeu~、オナリル
6 &6FVfv~ワカニヨワ
7 '7GUgu~ハナヲワ
8 (8HXhx~! ｾｸﾅﾘﾙ
9 )9IYiy~! ｾｸﾅﾘﾙ
a *:JZjz~! ｾｸﾅﾘﾙ
b +;Kck~! ｾｸﾅﾘﾙ
c ,<L¥!~! ｾｸﾅﾘﾙ
d -=Mjm~! ｾｸﾅﾘﾙ
e .>N^n~! ｾｸﾅﾘﾙ
f /?0_o + ｾｸﾅﾘﾙ
Ok

```

## 20H：8251からデータを入力する。

ポ ー ト	20H
-------	-----

I / O	INPUT
-------	-------

機 能	8251からデータを入力する。
-----	-----------------

解 説	8251 (USART) からのデータを受信するための入力ポートです。
-----	-------------------------------------

詳しくは8251 (USART) のデータ・シート等を御参照ください。

## 20H：8251へデータを出力する。

ポ ー ト	20H
-------	-----

I / O	OUTPUT
-------	--------

機 能	8251へデータを出力する。
-----	----------------

解 説	8251 (USART) へデータを送信するための出力ポートです。
-----	-----------------------------------

詳しくは8251 (USART) のデータ・シート等を御参照ください。

## 21H:8251からステータスを入力する.

ポ ー ト	21H
I / O	INPUT
機 能	8251からステータスを入力する.

**解 説** ステータスのリードは、入力ポート21H番地から入力する事によって行ないます.

詳しくは8251 (USART) のデータ・シートに譲りますが、参考のため8251 (USART) のステータスを示します.

第7ビット (TxRDY) ……このビットが1にセットされている場合には8251 (USART) の送信バッファが空になっている事を示し、次の送信データをバッファにセットする事が可能になります.

第6ビット (RxRDY) ……8251 (USART) が受信したシリアル・データをパラレル・データに変換し終え、CPUに渡せる状態にあるときセットされます.

第5ビット (Tx E) ……8251 (USART) は送信バッファのデータを、パラレル・シリアル変換回路によってシリアル・データに変換した上で外部に送り出しています. 送信バッファのデータを変換回路にセットした時点でバッファが空になりTxRDYがセットされますが、この時点ではデータ全てが外部に送り出されたわけではありません.

全てのデータが外部に送り出されてはじめてTx Eがセットされ、全てのデータ送信が終了した事を確認できます.

第4ビット (PE) ……パリティ・エラーの検出でセットされます.

第3ビット (OE) ……次のデータが準備されているにもかかわらず、CPUが前のデータをリードしない場合にセットされます.

第2ビット (FE) ……有効ストップ・ビットが検出されない場合にセットされます.

第1ビット (SYNDET) ……同期通信でのみ使用します.

第0ビット (DSR) ……モデム等の制御信号で、データ通信可能な状態であるか否かを示します.



## 21H: 8251へコントロール・ワードを出力する.

ポ ー ト	21H
I / O	OUTPUT
機 能	8251へコントロール・ワードを出力する.

**解 説** 8251 (USART) にはモード設定およびコマンド設定の2種類が必要ですが、モード設定はハードまたはコマンドによってリセットされた直後に1回だけ行なう事が可能で、以後21H番地に出力したものは全てコマンドとして受け取られます。

ところで、ユーザーのプログラム中から8251 (USART) を制御する場合には8251がリセット直後なのか、すでにモード設定が行なわれた後なのかを知る事ができません。そこで8251 (USART) をリセットしたい場合には1度ダミーのデータを送って完全にコマンド待ちの状態にしてからリセット・コマンドを送信します。

8251 (USART) 制御に関する明詳はデータ・シート等に譲りますが、モード設定、コマンド設定にわけて各ビットの意味を示します。

モード設定における各ビットの意味を示します。

第7ビット (S<sub>2</sub>) および第6ビット (S<sub>1</sub>) ……ストップ・ビットの長さを指定するために使用します。00のときストップ・ビット無効, 01のとき1ビット分, 10のとき1.5ビット分, 11のとき2ビット分のストップ・ビットが指定されます。

第5ビット (EP) ……パリティ・ビットを付ける場合に、偶数 (EVEN) か奇数 (ODD) かの指定を行ないます。

第4ビット (PEN) ……1のときパリティ・イネーブル, 0のときディスエーブルとなります。

第3ビット (L<sub>2</sub>) および第2ビット (L<sub>1</sub>) ……転送するデータのキャラクタ長を指定します。00のとき5ビット長, 01のとき6ビット長, 10のとき7ビット長, 11のとき8ビット長を指定しますが、N<sub>88</sub>-BASICおよびN-BASICではJIS 8単位を使用していますから8ビット必要です。

第1ビット (B<sub>2</sub>) および第0ビット (B<sub>1</sub>) ……両者共に0の場合は同期通信モードを、それ以外のときは非同期通信モードを指定

しますが、非同期通信モードの場合には同時に転送速度（ボーレート）も指定できます。01のとき1、10のとき16、11のとき64となりますが、転送速度は8251（USART）に入力されているクロックの周波数と次のような関係があります。

$$\text{転送速度（ボーレート）} = \frac{\text{周波数}}{1 \text{ または } 16 \text{ または } 64}$$

たとえば、クロックが4.8 KHz のときに第1ビット（B<sub>2</sub>）が1、第0ビット（B<sub>1</sub>）が0であれば、

$$\text{転送速度（ボーレート）} = \frac{4800}{16} = 300$$

で転送速度300ボーとなります。

コマンド設定における各ビットの意味を示します。

第7ビット（EH）…………同期通信の場合に使用するビットで、HUNTモードで1にセットした場合にはSYNCキャラクタの検出をはじめます。

第6ビット（IR）…………ソフトウェア・リセットを行ないます。

第5ビット（RTS）…………センド要求のためのRTS端子をコントロールするためのビットで、1にセットすると端子が0になります。

第4ビット（ER）…………ステータスのエラー・フラグ全てをリセットするためのビットで、PE、OEおよびFEをリセットします。

第3ビット（SBRK）…………通常は0にしておきますが、TxDの送信データを0にする場合のみ1にセットします。

第2ビット（RXE）…………受信イネーブルの指定に使用し、1のときイネーブル、0のときディスエーブルとなります。

第1ビット（DTR）…………データ・ターミナル・レディの意で、DTR端子をコントロールします。

第0ビット（TXEN）…………送信イネーブルの指定を行ないます。TXENのビットを1にセットしておけば、CDS端子が0になる事で送信可能（イネーブル）になり、CDS端子が0になっていれば、TXENをイネーブルにする事で送信可能になります。

### 30H:ディップ・スイッチのステータスを入力する.

ポ ー ト	30H
I / O	INPUT
機 能	ディップ・スイッチのステータスを入力する.

**解 説** 入力ポート30H番地は、上位2ビットが汎用の入力ポートとして、下位6ビットがPC-8801後面のディップ・スイッチ(SW1)のステータス入力用として用いられます。

PC-8801には4ビットの汎用入力ポートと1ビットの汎用出力ポートが用意されていますが、その汎用入力ポートの中の2ビットが30H番地の第7ビットと第6ビットに、残りの2ビットは31H番地の第7ビットと第6ビットに接続されています。ちなみに汎用出力ポートは40H番地に接続されており、これらの汎用ポートはユーザーが自由に使用できます。

下位6ビットからは、SW1のステータスを入力することができますが、以下に各ビットとディップ・スイッチの関係を示します。

第5ビット………SW1-6. このスイッチはN88-BASICでは使用されていません。

第4ビット………SW1-5. このスイッチはN88-BASICでは使用されていません。

第3ビット………SW1-4. ON(0)のとき25行モード、OFF(1)のとき25行モードとなります。

第2ビット………SW1-5. ON(0)のとき80桁モード、OFF(1)のとき40桁モードとなります。

第1ビット………SW1-4. ON(0)のときターミナル・モード、OFF(1)のときBASICモードとなります。

第0ビット………SW1-5. ON(0)のときN-BASICモード、OFF(1)のときN88-BASICモードとなります。

これらのビットは全て、ディップ・スイッチがON(上向き)のときビットが0、スイッチがOFF(下向き)のときビットが1になります。

### 30H：各種のコントロール信号を出力する。

ポ ー ト	30H
I / O	OUTPUT
機 能	各種のコントロール信号を出力する。

**解 説** 30H番地の出力ポートは主としてシリアル・インタフェースおよびCRTのコントロールに使用されます。以下に各ビットの意味を示します。

第7ビット……………使用していません。

第6ビット……………使用していません。

第5ビット (CB2) ……プログラマブル・コミュニケーション・インタフェース8251をSIOに使用するかCMTに使用するか  
の指定を行ないます。1の時にはSIOとなります。

第4ビット (CB1) ……USART8251をCMTに使用する  
場合の転送速度(ボーレート)を指定するためのビットで1の時に  
高速(SIOの場合は同期)となります。

第3ビット (MOTOR) ……本体に内蔵されているCMT用のモータ  
・リレーをコントロールするためのビットで1でON、0でOFF  
となります。

第3ビット (CINH) ……CMTインタフェースは、マーク2400  
Hz、スペース1200HzのFSK方式になっていますが、こ  
のビットを0にリセットしておいた場合にはCMTへ出力した  
データに対応した信号が出され、1にセットしておけば常に1  
200Hzの信号が出力されます。

第1ビット (COLOR) ……カラー・モードと白黒モードを切り換  
えるためのビットで、外部回路のみを設定する事ができます。

第0ビット (80) ……CRTコントローラではイニシャライズ時にス  
クリーンに表示する桁数を設定できますが、PC-8801で  
はCRTコントローラは常に80桁モードで使用し80/40  
桁の切り換えは、外部のタイミング回路によって行ないます。  
この回路を切り換えるためのビットが第0ビットです。

なお、N88-BASICでは、30H番地の出力ポートに出力したデータをE  
6C0H番地に保存してあります。

## 3 1 H：ディップ・スイッチのステータスを入力する。

ポ ー ト	3 1 H
I / O	INPUT
機 能	ディップ・スイッチのステータスを入力する。

**解 説** 入力ポート 3 1 H 番地は、上位 2 ビットが汎用の入力ポートとして、下位 6 ビットが PC-8801 後面のディップ・スイッチ (SW2) のステータス入力用として用いられます。

PC-8801 には 4 ビットの汎用入力ポートと 1 ビットの汎用出力ポートが用意されていますが、その汎用入力ポートの中の 2 ビットが 3 1 H 番地の第 7 ビットと第 6 ビットに、残りの 2 ビットは 3 0 H 番地の第 7 ビットと第 6 ビットに接続されています。ちなみに汎用出力ポートは 4 0 H 番地に接続されており、これらの汎用ポートはユーザーが自由に使用できます。

下位 6 ビットからは、SW2 のステータスを入力する事ができますが、以下に各ビットとディップ・スイッチの関係を示します。

第 5 ビット………SW2-6. ON(0)のとき半二重モード, OFF(1)  
のとき全二重モードとなります。

第 4 ビット………SW2-5. ON(0)のとき X パラメータ有効, OFF  
(1)のとき X パラメータ無効となります。

第 3 ビット………SW2-4. ON(0)のときストップ・ビット 1 ビット,  
OFF(1)のときストップ・ビット 2 ビットとなります。

第 2 ビット………SW2-3. ON(0)のとき 8 ビット長, OFF(1)の  
とき 7 ビット長となります。

第 1 ビット………SW2-2. ON(0)のときパリティ有り, OFF(1)  
のときパリティ無しとなります。

第 0 ビット………SW2-1. ON(0)のとき奇数 (ODD) パリティ,  
OFFのとき偶数 (EVEN) パリティとなります。

これらのビットは全て、ディップ・スイッチが ON (上向き) のときビットが O, スイッチが OFF (下向き) のときビットが 1 になります。

### 3 1 H : C R T およびメモリのモード設定を行なう。

ポ ー ト	3 1 H
-------	-------

I / O	OUTPUT
-------	--------

機 能	CRTおよびメモリのモード設定を行なう。
-----	----------------------

**解 説** 出力ポート 3 1 H 番地は、メモリ・モードおよび CRT のディスプレイ・モードをコントロールするためのポートで PC-8801 ならではの大切な出力ポートです。

メモリ・モードは、N88-BASICモード、N-BASICモード、64Kバイト・オールRAMモードの3モードを指定する事ができます。

上位2ビットは使用されていませんが、メモリ・モードの選択は第2ビットおよび第1ビットにより、CRTディスプレイ・モードの選択は第5ビット、第4ビット、第3ビット、および第0ビットによって行ないます。

以下に各ビットの意味を示します。

第5ビット……水平周波数24.8KHzの専用高解度ディスプレイ(PC-8851またはPC-8853等)を使用したハイスピードCRTモードにおいて、0のとき25行モードを、1のとき20行モードを指定します。

第4ビット……第3ビットでグラフィックを指定した場合において、この第4ビットが0のとき通常の白黒グラフィック・モード、1のときカラー・グラフィック・モードとなります。

第3ビット……0のときテキスト・スクリーンのみを、1のときにはグラフィックを使用できます。

第2ビット……0のときN88-BASICモード、1のときN-BASICモードとなります。

第1ビット……1のとき32Kバイト・オールRAMモードとなり、0のときには第2ビットでモード指定を行ないます。

第0ビット……ハイスピードCRT(専用高解像度ディスプレイ)モードのグラフィックにおいて、0のとき640×400ドット×1プレーンを、1のとき640×200ドット×3プレーンを指定します。



## 40H：各種のコントロール信号を入力する。

ポ ー ト	40H
I / O	INPUT

機 能 各種のコントロール信号を入力する。

**解 説** 入力ポート40H番地のコントロール機能は、CRT、タイマ、拡張ユニット、シリアル・インタフェース、プリンタと多岐にわたっています。以下に各ビットの意味を示します。

第7ビット……………使用していません。

第6ビット……………使用していません。

第5ビット (VRTC) ……CRTのコントロールに使用します。0のとき表示および水平リトレース期間、1のときはバーチカル・リトレース期間です。

第4ビット (DATA IN) ……タイマからの入力を行なうためのビットで、出力ポートのCLKと同期させてシリアル・データを受け取ります。

第3ビット (EXTON) ……外部バスに接続されているEXTON信号の状態を示します。

第2ビット (CDIN) ……シリアル・インタフェース(SIO)のキャリア・ディテクト検出に使用し、CMTからのキャリア信号を検出した場合1にセットされます。

第1ビット (HCRT) ……0の場合ハイ・スピードCRTモード(専用高解像度ディスプレイ使用で600×400ドット可能)、1の場合ノーマルCRTモードです。このビットはPC-8801後面のジャンパ・スイッチの状態によって変化します。

第0ビット (BUSY) ……プリンタからのBUSY (READY) 信号の検出に使用します。



## 40H：各種のコントロール信号を出力する。

ポ ー ト	40H
I / O	OUTPUT
機 能	各種のコントロール信号を出力する。

**解 説** 出力ポート40H番地のコントロール機能は、ブザー、CRT、タイマ、プリンタと多岐にわたっている他に、1ビットの汎用出力ポートも有しています。

以下に各ビットの意味を示します。

第7ビット……………使用していません。

第6ビット (UOP0) ……1ビットの汎用出力ポートとして利用する事ができます。

第5ビット (BEEP) ……内蔵ブザーをコントロールするためのビットで、1にセットする事でブザーを鳴らすことができます。回路上ではこのビットからの出力信号と、2.4KHzのクロック信号との論理積 (AND) がスピーカに出力されます。

第4ビット (FLASH) ……このビットをセットする事によってフラッシング (高速書き込み) モードとなります。このモードでは出力ポート5CH～5EH番地の制御によって、いずれかのGVRAMをセレクトした場合に自動的に表示されなくなり、実行速度がおそくなるのを防げます。

第3ビット (CLDS) ……CRTコントローラをイニシアライズする際の同期コントロールで0がアクティブです。

第2ビット (CLK) ……タイマとの入出力をコントロールするためのシフト・クロックを与えます。

第1ビット (CSTB) ……タイマにコマンドをセットするために、ストローブ信号を与えます。

第0ビット (PSTB) ……プリンタヘストローブ信号を与えるために使用します。初期状態では1にセットしておきませんが、1→0→1と変化させる事によってストローブ信号を与える事ができます。

なお、N88-BASICでは、40H番地の出力ポートに出力したデータをE6C1H番地に保存してあります。

## 50H：CRTコントローラへパラメータを出力する。

ポ ー ト	50H
I / O	OUTPUT
機 能	CRTコントローラへパラメータを出力する。

**解 説** PC-8801が使用している $\mu$ PD3301は、同期信号発生器、行バッファ・メモリ、アトリビュート記憶メモリ等を1チップ上に搭載したプログラマブルCRTコントローラで、カラー表示と白黒表示ができ、かつ表示フォーマットがプログラマブルであるため、行数、桁数の選定が任意であるなどの特長を有しています。アトリビュートに関しては、文字コードとアトリビュートを混在して使用するノン・トランスペアレント・アトリビュートと、文字コード・エリアとアトリビュート・エリアとに分離したトランスペアレント・アトリビュートを使用する事が可能で、アトリビュートを設けない事もできます。

ノン・トランスペアレント・アトリビュートは、文字コードとアトリビュート・コードを同一のVRAM上に混在する方法で両者の区別を最上位ビット(MSB)によって行なう非常に合理的な方法ではあるのですが、PC-8801では256種のキャラクタを用いるためにトランスペアレント・アトリビュートを採用しています。

このCRTコントローラにパラメータを設定するための出力ポートが50H番地です。CRTコントローラ(3301)に与えるコマンドおよびパラメータの実際の意味を知りたい方は、データ・シート等を御参照ください。

## 51H：CRTコントローラへコマンドを出力する。

ポ ー ト	51H
I / O	OUTPUT
機 能	CRTコントローラへパラメータを出力する。

**解 説** 上記のCRTコントローラ(3301)にコマンドを設定するための出力ポートが51H番地です。

詳しくは、 $\mu$ PD3301のデータ・シート等を御参照ください。

## 5 2 H : ボーダー・カラー等を設定する.

ポ ー ト	5 2 H
I / O	OUTPUT
機 能	ボーダー・カラーおよびバック・グラウンド・カラーを設定する.

**解 説** 5 2 H番地の出力ポートは、CRTスクリーンのボーダー・カラー（テキスト表示エリアの外枠の色）およびバック・グラウンド・カラー（グラフィック画面の背景の色）を制御するためのポートです。

以下に、5 2 H番地に出力するデータの各ビットと、ボーダー・カラー、バック・グラウンド・カラーの関係を示します。

第7ビットは無関係です。

第6ビットでバック・グラウンド・カラーのGREEN信号を制御します。

第5ビットでバック・グラウンド・カラーのRED信号を制御します。

第4ビットでバック・グラウンド・カラーのBLUE信号を制御します。

第3ビットは無関係です。

第2ビットでボーダー・カラーのGREEN信号を制御します。

第1ビットでボーダー・カラーのRED信号を制御します。

第0ビットでボーダー・カラーのBLUE信号を制御します。

この、ボーダー・カラーおよびバック・グラウンド・カラーは、N<sub>88</sub>-BAS I Cとは無関係な、たとえばN-BAS I Cモードや6 4 Kオール・ラムのモードにおいても利用する時ができます。

### サンプル

```
100 /  
110 / execute on n88-basic mode  
120 /  
130 SCREEN 1  
140 FOR BO=7 TO 0 STEP -1  
150   FOR BK=7 TO 0 STEP -1  
160     OUT &H52,BO*&H10+BK  
170     FOR W=1 TO 100:NEXT W  
180   NEXT BK  
190 NEXT BO
```

```
100 /  
110 / execute on pc-8801 n-basic mode  
120 /  
130 /  
140 FOR BO=7 TO 0 STEP -1  
150   FOR BK=7 TO 0 STEP -1  
160     OUT &H52,BO*&H10+BK  
170     FOR W=1 TO 100:NEXT W  
180   NEXT BK  
190 NEXT BO
```

## 5 3 H : 表示する G V R A M を指定する.

ポ ー ト	5 3 H
I / O	OUTPUT
機 能	表示する G V R A M を指定する.

**解 説** 出力ポート 5 3 H 番地は下位 (L S B) 4 ビットが有効で、最下位ビット (L S B) によってテキスト画面を表示するか否かを指定することができます。なお、5 3 H 番地のポートは全ビットがアクティブ・ロウ (LOW) であるため、最下位ビットも 1 の時テキスト画面を表示しません (D I S A B L E)。通常はテキスト画面を表示するために、最下位ビットは 0 にしておきます。

第 3 ビット～第 1 ビットは、グラフィック・モードにおいてのみ有効です。

グラフィック・モードにおいては、6 4 0 × 2 0 0 ドット白黒のグラフィック画面を 3 画面持っていますが、このモードにおいては G V R A M 0、G V R A M 1、G V R A M 3 のグラフィック用 V R A M を独立して使用しています。

これら 3 組の V R A M を表示するか否かを指定するビットが第 3 ビット～第 1 ビットで、対応は次のようになっています。

- 第 3 ビットが 1 の場合、G V R A M 2 を表示しません (D I S A B L E)。
- 第 2 ビットが 1 の場合、G V R A M 1 を表示しません (D I S A B L E)。
- 第 1 ビットが 1 の場合、G V R A M 0 を表示しません (D I S A B L E)。
- 第 0 ビットが 1 の場合、テキスト画面を表示しません (D I S A B L E)。

## 5 4 H～5 B H：カラー・パレットを設定する。

ポ ー ト	5 4 H～5 B H
I / O	OUTPUT
機 能	カラー・パレットを設定する。

**解 説** N88-BASICのカラー・グラフィック・モードでは、グラフィック命令においてパレット番号が指定できるようになっており、これでカラー指定が行なえます。また、このパレット番号を省略した場合には、フォア・グラウンド・カラーとして指定した値がパレット番号として採用されます。

を変える事によって瞬時に変更する事ができますが、このカラー・パレットが5 4 H～5 B H番地の出力ポートに用意されています。

5 4 H～5 B H番地がカラー・パレット0～パレット7に対応しており、これらのポートは下位3ビットのみが有効です。

出力ポート5 4 H番地はカラー・パレット0です。

出力ポート5 5 H番地はカラー・パレット1です。

出力ポート5 6 H番地はカラー・パレット2です。

出力ポート5 7 H番地はカラー・パレット3です。

出力ポート5 8 H番地はカラー・パレット4です。

出力ポート5 9 H番地はカラー・パレット5です。

出力ポート5 A H番地はカラー・パレット6です。

出力ポート5 B H番地はカラー・パレット7です。

それぞれの下位3ビットは次の意味を持ちます。

第2ビットをセットするとパレットのGREEN信号がONになります。

第1ビットをセットするとパレットのRED信号がONになります。

第0ビットをセットするとパレットのBLUE信号がONになります。

## サンプル

```

100 /
110 / color palette demonstration fast version
120 /
130 CLEAR ,&HB8FF
140 SCREEN 0,3:WIDTH 40,20:CONSOLE 0,20,0,1
150 COLOR 7:CLS 1:LOCATE 5,7:PRINT 'color palette demonstration'
160 CLS 2:SCREEN 0,0:FOR T=0 TO 1000:NEXT T:CLS 1
170 R=120:N=35:PI=3.14159
180 GOSUB *SUBCOLOR
190 GOSUB *SUBTEXT
200 GOSUB *SUBGRAPHIC
210 GOSUB *SUBPALETTE
220 GOSUB *SUBCOLOR
230 LOCATE 0,0
240 END
250 /
260 *SUBTEXT
270 LOCATE 16,9:PRINT 'NEC'
280 LOCATE 17,10:PRINT 'PC-8801'
290 RETURN
300 /
310 *SUBGRAPHIC
320 FOR GR=1 TO N
330   A=GR*2*PI/N:X=2*R*COS(A)+319:Y=R*SIN(A-PI/4)/2+99
340   C=C+1:COL=(C MOD 7)+1
350   CIRCLE(X,Y),30,COL
360   LINE -(319,99),COL
370 NEXT GR
380 RETURN
390 /
400 *SUBPALETTE
410 RESTORE
420 FOR ADDR=&HB900 TO &HB928
430   READ CODE$:POKE ADDR,VAL('&h'+CODE$)
440 NEXT ADDR
450 ADDR=&HB900:CALL ADDR
460 RETURN
470 /
480 DATA 1e,ff      t'      ld      e,0ffh
490 DATA 16,55      t' pal:  ld      d,55h
500 DATA 0e,55      t' pal2: ld      c,55h
510 DATA 7a         t' pal3: ld      a,d
520 DATA b9         t'      cp      c
530 DATA 20,04      t'      jr      nz,notdsp
540 DATA 3e,07      t'      ld      a,7
550 DATA 18,01      t'      jr      dsplay
560 DATA af         t'      notdsp:xor a
570 DATA ed,79      t'      dsplay:out (c),a
580 DATA 0c         t'      inc     c
590 DATA 79         t'      ld      a,c
600 DATA fe,5c      t'      cp      5bh+1
610 DATA 20,ef      t'      jr      nz,pal3
620 DATA 21,e8,03   t'      ld      hl,1000
630 DATA 2b         t'      wait:  dec     hl
640 DATA 7c         t'      ld      a,h
650 DATA b5         t'      or      l
660 DATA 20,fb      t'      jr      nz,wait
670 DATA 14         t'      inc     d
680 DATA 7a         t'      ld      a,d
690 DATA fe,5c      t'      cp      5bh+1
700 DATA 20,df      t'      jr      nz,pal2
710 DATA 1d         t'      dec     e
720 DATA 20,da      t'      jr      nz,pal
730 DATA c9         t'      ret
740 /
750 *SUBCOLOR
760 FOR I=0 TO 7
770   COLOR=(I,I)
780 NEXT I
790 RETURN

```

## 5CH: GVRAMのステータスを入力する。

ポ ー ト	5CH
-------	-----

I / O	INPUT
-------	-------

機 能	GVRAMのステータスを入力する。
-----	-------------------

**解 説** 入力ポート5CH番地からの入力データは、GVRAM0、GVRAM1、GVRAM2の3バンクのGVRAMの中でどのGVRAMがセレクトされている（ENABLE）のか、またセレクトされていない（DISABLE）状態であるのかのステータスを意味しています。

入力データは、第2ビット～第0ビットがそれぞれGVRAM2～GVRAM0に対応しており、上位5ビットは関係ありません。

第2ビットが1の場合GVRAM2がセレクトされています。

第1ビットが1の場合GVRAM1がセレクトされています。

第0ビットが1の場合GVRAM0がセレクトされています。

つまり、通常のメインRAMがセレクトされている状態では、第2ビット～第0ビット全てが0にリセットされています。



## 5CH～5FH：GVRAMをセレクトする。

ポ ー ト	5CH～5FH
I / O	OUTPUT
機 能	GVRAMをセレクトする。

**解 説** PC-8801では、16KバイトのGVRAM（グラフィック専用VRAM）を3バンク標準実装していますが、これらのGVRAMは通常CPUのアドレス・マップ上にはありません。

そこで、GVRAMをCPUから直接アクセスできるようにするためのバンク切り換えを行なえなければいけません、このバンク切り換えを指定するための出力ポートが5CH～5FH番地のポートです。

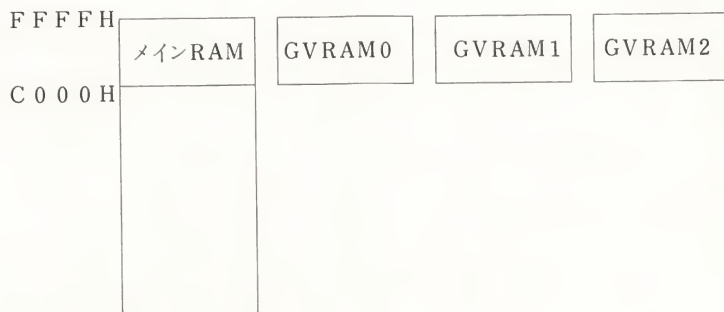
GVRAM0～GVRAM2および通常N88-BASICのワーク・エリア等として用いているメインRAMの、4つのバンクが、5CH～5FH番地の4つの出力ポートに対応しており、それぞれの出力ポートに出力データ（データ自身は意味を持たない）を出力する事によって各バンクをセレクトする事ができます。セレクトしたRAMはCPUのマップ上C000H～FFFFH番地に位置します。

出力ポート5CH番地へ出力した場合GVRAM0をセレクトします。

出力ポート5DH番地へ出力した場合GVRAM1をセレクトします。

出力ポート5EH番地へ出力した場合GVRAM2をセレクトします。

出力ポート5FH番地へ出力した場合メインRAMをセレクトします。



# サンプル

```

100 /
110 / test graphic video ram
120 /
130 CLEAR ,&HB8FF
140 FOR ADDR=&HB900 TO &HB94A
150 READ CODE$:POKE ADDR,VAL('&h'+CODE$)
160 NEXT ADDR
170 SCREEN 0,0
180 FOR I=0 TO 7
190 COLOR=(I,I)
200 NEXT I
210 ADDR=&HB900:CALL ADDR
220 END
230 /
240 DATA f3 : di
250 DATA ed,73,fe,b9 : ld (0b9feh),sp
260 DATA 31,fe,b9 : ld sp,0b9feh
270 DATA 06,01 : ld b,1
280 DATA cd,2f,b9 : loop: call clr
290 DATA 0e,ff : ld c,0ffh
300 DATA cb,40 : bit 0,b
310 DATA c4,37,b9 : call nz,gvram0
320 DATA cb,48 : bit 1,b
330 DATA c4,3b,b9 : call nz,gvram1
340 DATA cb,50 : bit 2,b
350 DATA c4,3f,b9 : call nz,gvram2
360 DATA 04 : inc b
370 DATA 78 : ld a,b
380 DATA fe,08 : cp 7+1
390 DATA 20,e6 : jr nz,loop
400 DATA cd,2f,b9 : call clr
410 DATA d3,5f : out (5fh),a
420 DATA ed,7b,fe,b9 : ld sp,(0b9feh)
430 DATA eb : ei
440 DATA c9 : ret
450 /
460 DATA 0e,00 : clr: ld c,0
470 DATA cd,3f,b9 : call gvram2
480 DATA cd,3b,b9 : call gvram1
490 DATA d3,5c : gvram0:out (5ch),a
500 DATA 18,06 : jr full
510 DATA d3,5d : gvram1:out (5dh),a
520 DATA 18,02 : jr full
530 DATA d3,5e : gvram2:out (5eh),a
540 DATA 21,00,c0 : full: ld hl,0c000h
550 DATA 71 : full1: ld (hl),c
560 DATA 23 : inc hl
570 DATA 7c : ld a,h
580 DATA b5 : or l
590 DATA 20,fa : jr nz,full1
600 DATA c9 : ret
610 /

```

## 60H～68H：DMAコントローラを制御する。

ポ ー ト	60H～68H
I / O	OUTPUT (68H番地のみ INPUT/OUTPUT)
機 能	DMAコントローラを制御する。

**解 説** DMAコントローラ ( $\mu$ PD8257) は、テキスト用VRAMのデータをCRTコントローラ ( $\mu$ PD3301) へDMA転送するために使用されています。

$\mu$ PD8257はプログラマブルなDMAコントローラで、4チャンネルに対して、ソース・アドレスおよび転送容量を指定する事ができますが、N88-BASICではオート・ロード・モードを使用しているためチャンネル2しか用いる事ができません。オード・ロード・モードでは、一重直画面ごとに自動的にDMAアドレスのロードおよびプリセットが行なわれています。

以下に、DMAコントローラ制御ポートを示します。

60H番地	.....チャンネル0	DMAアドレス・セット
61H番地	.....チャンネル0	ターミナル・カウンタ・セット
62H番地	.....チャンネル1	DMAアドレス・セット
63H番地	.....チャンネル1	ターミナル・カウンタ・セット
64H番地	.....チャンネル2	DMAアドレス・セット
65H番地	.....チャンネル2	ターミナル・カウンタ・セット
66H番地	.....チャンネル3	DMAアドレス・セット
67H番地	.....チャンネル3	ターミナル・カウンタ・セット
68H番地	.....モード・セットおよびステータス・リード	

N88-BASICではこれらのポートの中でチャンネル2関係の64H番地、65H番地、および68H番地を使用します。68H番地のみはステータス・リードのために入力ポートとして使用する事ができます。

チャンネル0およびチャンネル1は外部にサポートされていますが、8インチDMA転送方式のフロッピー・ディスク・ユニットを接続する事によって1チャンネルは専有されてしまいます。

PC-8801のスロット・バスには、 $\overline{\text{DRQ1}}$ 、 $\overline{\text{DRQ2}}$ 、 $\overline{\text{DACK1}}$ 、 $\overline{\text{DACK2}}$ 、 $\overline{\text{DMATC}}$ 、 $\overline{\text{FDRDY}}$ 、それぞれの信号がサポートされており、I/O機器との間でDMA転送が行なえる様になっています。

## 70H：オフセット・アドレスを入力する。

ポ ー ト	70H
I / O	INPUT
機 能	オフセット・アドレスを入力する。

**解 説** N88-BASICはテキスト・エリアとして0000H番地から7FFFH番地のRAM32Kバイトを使用しますが、このエリアは通常CPUのメモリ・マップ上にはありません。このためN88-BASICからテキスト・エリアのデータを読む場合、またテキスト・エリアへデータを書き込む場合にはハードウェアによって、テキスト・エリア上の任意の1Kバイトを8000H番地から83FFH番地のテキスト・ウィンドウにマップし、このウィンドウに対して読み書きを行ないます。

PC-8801では、このテキスト・ウィンドウへのマッピングを行なうためにオフセット・アドレス・レジスタを持っていますが、オフセット・アドレス・レジスタの値は70H番地の入力ポートをリードする事によって調べられます。

具体的には、8000H～83FFH番地がアクセスされたとき、アクセスされたアドレスの最上位ビット(MSB)をマスクします。次にMSBをマスクしたアドレスとオフセット・アドレス・レジスタのオフセット・アドレスを加えたものを実際のアドレスとしてRAMのアクセスを行ないます。

以上の機能により、オフセット・アドレス・レジスタの内容を変化させる事によってテキスト・ウィンドウを通してアクセスする事の可能なアドレスを256バイト単位で変化させる事ができます。

ただし、0000H～7FFFH番地のRAMは常に書き込みだけはサポートされています。

## 70H：オフセット・アドレス・レジスタを設定する。

ポ ー ト	70H
-------	-----

I / O	OUTPUT
-------	--------

機 能	オフセット・アドレス・レジスタを設定する。
-----	-----------------------

解 説	オフセット・アドレス・レジスタにデータを書き込むための出力ポートです。
-----	-------------------------------------

CPUから8000H～83FFH番地がアクセスされた場合には、アクセスされたアドレスの最上位ビット（MSB）マスクし、さらにオフセット・アドレス・レジスタのオフセット・アドレスを加えたものを実際のアドレスとしてRAMをアクセスします。

## 71H：拡張ROMのステータスを入力する.

ポ ー ト	71H
I / O	INPUT
機 能	拡張ROMのステータスを入力する.

**解 説** PC-8801では、6000H~7FFFH番地のROMエリアに対して8バンクの増設をサポートしています。1バンクはN88-BASICのインタプリタみずからがグラフィック制御ルーチン等を置いて使用していますから、ユーザーが自由に使う事のできるのは7バンクのみでスロット内蔵設可能となっています。

これらの拡張ROMの中でどのROMがセレクトされているか、またはメインROMがセレクトされているのかを示すステータスが入力ポート71H番地に入力しています。

次に入力ポート71H番地の各ビットと拡張ROMセレクトの関係を示します。

第7ビットが0の場合、拡張ROM8がセレクトされています。

第6ビットが0の場合、拡張ROM7がセレクトされています。

第5ビットが0の場合、拡張ROM6がセレクトされています。

第4ビットが0の場合、拡張ROM5がセレクトされています。

第3ビットが0の場合、拡張ROM4がセレクトされています。

第2ビットが0の場合、拡張ROM3がセレクトされています。

第1ビットが0の場合、拡張ROM2がセレクトされています。

第0ビットが0の場合、拡張ROM1がセレクトされています。

以上のように、これらのビットはアクティブ・ロウ（LOW）で、全てのビットが1であった場合にのみ、N88-BASICメインROMの6000~7FFFH番地がセレクトされている事になります。

## 71H：拡張ROMをセレクトする。

ポ ー ト	71H
-------	-----

I / O	OUTPUT
-------	--------

機 能	拡張ROMをセレクトする。
-----	---------------

**解 説** PC-8801では、6000H～7FFFH番地のROMエリアに対して8バンクの増設をサポートしています。1バンクはN88-BASICのインタプリタみずからがグラフィック制御ルーチン等を置いて使用していますから、ユーザーが自由に使う事のできるのは7バンクのみでスロット内増設可能となっています。

これら拡張ROM8バンクおよびN88-BASICのメインROMをセレクトするのが出力ポート71H番地です。

次に、出力ポート71H番地の各ビットと拡張ROMセレクトの関係を示します。

第7ビットが0で拡張ROM8がセレクトされます。

第6ビットが0で拡張ROM7がセレクトされます。

第5ビットが0で拡張ROM6がセレクトされます。

第4ビットが0で拡張ROM5がセレクトされます。

第3ビットが0で拡張ROM4がセレクトされます。

第2ビットが0で拡張ROM3がセレクトされます。

第1ビットが0で拡張ROM2がセレクトされます。

第0ビットが0で拡張ROM1（N88で使用）がセレクトされます。

以上のように、これらのビットはアクティブ・ロウ（LOW）で、全てのビットが1であった場合にのみ、N88-BASICメインROMの6000H～7FFFH番地がセレクトされます。



## 78H：オフセット・アドレスをインクリメントする。

ポ ー ト 78H

I / O OUTPUT

機 能 オフセット・アドレスをインクリメントする。

**解 説** オフセット・アドレス・レジスタに設定されている、オフセット・アドレスをインクリメントするための出力ポートが78H番地です。

N88-BASICのモードにおいて、CPUから8000H～83FFH番地がアクセスされた場合には、アクセスされたアドレスの最上位ビット（MSB）をマスクし、さらにオフセット・アドレス・レジスタのオフセット・アドレスを加えたものを実際のアドレスとしてRAMをアクセスします。

オフセット・アドレス・レジスタの内容をインクリメントするためには、出力ポート78H番地への出力命令を実行すれば良く、出力データ自身は何でもかまいません。

また、オフセット・アドレス・レジスタへのリード・ライトは、入出力ポート70H番地を使用します。

### サンプル

```

;
; --- dump text ram area of n88-basic ---
;
; ORG 0B900H
B900 110000 ; LD DE,0000H ; clear address counter
;
B903 AF XOR A
B904 D370 OUT (70H),A ; clear offset register
;
B906 210080 LOOPLL:LD HL,8000H ; top address of text window
;
B909 0E10 LD C,16
B90B 7A LOOPL: LD A,D
B90C CD4FB9 CALL DSPACC ;
B90F 7B LD A,E ; display address
B910 CD4FB9 CALL DSPACC ;
B913 3E3A LD A,' '
B915 DF RST 18H
;
B916 E5 PUSH HL
B917 0610 LD B,16
B919 3E20 LOOPSL:LD A,' '
B91B DF RST 18H
B91C 7E LD A,(HL)
B91D CD4FB9 CALL DSPACC ; dump hexa decimal number
;
B920 23 INC HL
B921 10F6 DJNZ LOOPSL
B923 E1 POP HL
;
B924 CD64B9 CALL TENSPPC
;
B927 0610 LD B,16

```

```

B929 7E      LOOPS2:LD  A,(HL)
B92A FE20    CP      20H
B92C 3002    JR      NC,ASCII
B92E 3E2E    LD      A,'.'
B930 DF      ASCII: RST  18H      ; dump ascii character
;
B931 13      INC      DE
B932 23      INC      HL
B933 10F4    DJNZ    LOOPS2
;
B935 3E0A    LD      A,0AH
B937 DF      RST      18H
B938 3E0D    LD      A,0DH
B93A DF      RST      18H
;
B93B CDC235 CALL  35C2H      ; check stop key
B93E D8      RET      C      ; stop key, return to basic
;
B93F 0D      DEC      C
B940 20C9    JR      NZ,LOOPL
;
B942 D378    OUT      (78H),A      ;*increment offset register
;
B944 7A      LD      A,D      ;
B945 FE80    CP      80H      ; check end of msb byte
B947 20BD    JR      NZ,LOOPLL    ;
;
B949 7B      LD      A,E      ;
B94A FE00    CP      00H      ; check end of lsb byte
B94C 20B8    JR      NZ,LOOPLL    ;
;
B94E C9      RET      ; end of dump, return to basic
;
; output hexa decimal number in accumulator subroutine
;
; destroys : register a,f
;
B94F F5      DSPACC:PUSH AF
B950 0F      RRCA
B951 0F      RRCA
B952 0F      RRCA
B953 0F      RRCA
B954 CD58B9 CALL  HALF
B957 F1      POP      AF
;
B958 E60F    HALF: AND  0FH
B95A FE0A    CP      10
B95C 3802    JR      C,NUMBER
B95E C607    ADD      A,7
;
B960 C630    NUMBER:ADD A,30H
B962 DF      RST      18H
B963 C9      RET
;
; output ten spaces subroutine
;
; destroys : register a,f,b
;
B964 3E20    TENSPP:LD  A,' '
B966 060A    LD      B,10
B968 DF      TENSPP1:RST  18H
B969 10FD    DJNZ    TENSPP1
B96B C9      RET
;

```

```

a=&hb900:call a
0000: 00 09 00 0A 00 3A 8F E9 00 1F 00 14 00 3A 8F E9
0010: 20 46 6F 72 6D 61 74 20 61 20 64 69 73 68 00 27
0020: 00 1E 00 3A 8F E9 00 54 00 28 00 3A 8F E9 20 20
0030: 20 20 20 20 20 20 20 20 20 20 20 43 6F 70 79 72 69
0040: 67 68 74 20 28 43 29 20 31 39 38 32 20 62 79 20
0050: 4E 45 43 00 5C 00 32 00 3A 8F E9 00 66 00 3C 00
0060: AB 20 41 F4 5A 00 6E 00 46 00 3A 8F E9 00 7C 00
0070: 50 00 53 59 53 54 52 48 35 F1 14 00 87 00 5A 00
0080: 9D 20 2C 2C 11 00 8F 00 64 00 CB 20 11 00 DB
0090: 00 6E 00 91 20 22 4D 6F 75 6E 74 20 61 20 22 3B
00A0: 3A CB 20 15 3A 91 22 6E 65 77 22 3B 3A CB 20 11
00B0: 3A 91 22 20 64 69 73 68 20 6F 6E 20 22 3B 3A CB
00C0: 20 15 3A 91 22 65 76 65 6E 22 3B 3A CB 20 11 3A
00D0: 91 22 20 64 72 69 76 65 23 22 00 13 01 78 00 85
00E0: 20 22 46 6F 72 6D 61 74 20 64 72 69 76 65 23 22
00F0: 3B 44 52 49 56 45 3A 20 8B 20 44 52 49 56 45 20
0100: FD 20 13 20 F1 20 12 20 0D 20 D7 3A 20 89 20 0E
0110: 78 00 00 1B 01 82 00 3A 8F E9 00 2A 01 8C 00 44
0120: 52 56 54 42 4C F1 0C 64 EF 00 41 01 96 00 4D 41
0130: 58 54 52 4B F1 FF D0 28 44 52 49 56 45 2C 11 29
0140: 00 58 01 A0 00 4D 41 58 53 45 43 F1 FF D0 28 44
0150: 52 49 56 45 2C 12 29 00 6F 01 AA 00 44 4F 55 42
0160: 4C 45 F1 FF D0 28 44 52 49 56 45 2C 13 29 00 86
0170: 01 B4 00 43 4C 53 54 52 4B F1 FF D0 28 44 52 49
0180: 56 45 2C 14 29 00 9F 01 BE 00 4D 41 58 43 4C 53
0190: F1 FF D0 28 44 52 49 56 45 2C 15 29 F4 12 00 B6
01A0: 01 C8 00 44 49 52 54 52 4B F1 FF D0 28 44 52 49
01B0: 56 45 2C 16 29 00 CD 01 D2 00 53 45 43 43 4C 53
01C0: F1 FF D0 28 44 52 49 56 45 2C 17 29 00 E4 01 DC
01D0: 00 46 41 54 4F 4E 45 F1 FF D0 28 44 52 49 56 45
01E0: 2C 18 29 00 FB 01 E6 00 46 41 54 4C 53 54 F1 FF
01F0: D0 28 44 52 49 56 45 2C 19 29 00 15 02 F0 00 4E
0200: 55 4D 4F 46 43 4F 50 59 F1 FF D0 28 44 52 49 56
0210: 45 2C 1A 29 00 2D 02 FA 00 44 53 4B 49 4E 46 F1
0220: FF D0 28 44 52 49 56 45 2C 0F 0A 29 00 47 02 04
0230: 01 4E 55 4D 4F 46 4D 4F 44 F1 FF D0 28 44 52 49
0240: 56 45 2C 0F 0B 29 00 68 02 0E 01 86 20 43 4C 55
0250: 53 54 45 52 25 28 4D 41 58 43 4C 53 29 2C 46 41
0260: 54 24 28 4D 41 58 43 4C 53 29 00 73 02 18 01 3A
0270: 8F E9 00 81 02 22 01 86 20 50 41 52 41 28 19 29
^C
Break
Ok

```

```

.....:.....:
Format a disk.
...:..T(..:..
Copyri
ght (C) 1982 by
NEC.¥.2.:+..f.<.
? ABZ.n.f.:+..!.
P.SYSTK5M..Z.
\.....+d..0
.n..T 'Mount a '
:..:.'new':..
:.. disk on ':..
:.'even':..
T' drive#...x.■
'Format drive#
;DRIVE: ■ DRIVE
. M . ン 7 : | .
x.....:..*.■.D
RVTLB..d.A.l.MA
XTRK ≡ (DRIVE,..)
.X. .MAXSEC ≡ (D
RIVE,..).o.i.DOUB
LE ≡ (DRIVE,..)■
.I.CLTRK ≡ (DRI
VE,..).'.e.MAXCLS
≡ (DRIVE,..)B..カ
.?.DIRTRK ≡ (DRI
VE,..).^..X.SECCLS
M ≡ (DRIVE,..).^..?
.FATONE ≡ (DRIVE
,..).^..FATLST ≡
(DRIVE,..)...X.N
UMOFCOPYM ≡ (DRIV
E,..).-. .DSKINF ≡
(DRIVE,..).G..
.NUMOFMOD ≡ (DRI
VE,..).k...■ CLU
STER%(MAXCLS),FA
T$(MAXCLS).s...:
+..-.'.■ PARA(..)

```

## E 4 H : 8 2 1 4 のカレント・レジスタを設定する。

ポ ー ト	E 4 H
I / O	OUTPUT
機 能	8 2 1 4 のカレント・レジスタを設定する。

**解 説** インタラプト・レベルを与える、インタラプト・コントローラ（8 2 1 4）のカレント・レジスタを設定するためのポートです。N88-BASICでは出力ポートE 4 H番地へ出力したインタラプト・レベルをE 6 C 3 H番地に保存しています。

## E 6 H : インタラプトをコントロールする。

ポ ー ト	E 6 H
I / O	OUTPUT
機 能	インタラプトをコントロールする。

**解 説** 出力ポートE 6 H番地の下位3ビットは、PC-8801においてサポートされている各種のインタラプトを可能（ENABLE）とするか禁止（DISABLE）するかのコントロールに使用します。

第2ビット………PC-8801のSIOは通常モードの他にインタラプト・モードを用いる事ができますが、この第2ビットに1を出力する事によってインタラプト・モードを選択できます。インタラプト・モードでは8251（USART）の受信バッファがレディになると、自動的にインタラプト要求信号を発生します。

第1ビット………VRTCの立ち下がりエッジを検出した場合に、レベル1のインタラプト要求信号を発生するか否かを選択するビットで、アクティブ・ハイ（ACTIVE HIGH）です。

第0ビット………リアル・タイム・インタラプトを使用するか禁止するかの指定をアクティブ・ハイ（ACTIVE HIGH）で行なうビットです。なお、N88-BASICではリアル・タイム・インタラプトを実際に使用しているため、ユーザが使う場合には十分な注意が必要です。

## E 8 H～E 9 H：漢字フォントを入力する。

ポ ー ト	E 8 H～E 9 H
-------	-------------

I / O	INPUT
-------	-------

機 能	漢字フォントを入力する。
-----	--------------

**解 説** PC-8801に漢字ROMポート(PC-8801-01)を実装した場合の、漢字フォントの読み込み手順を示します。

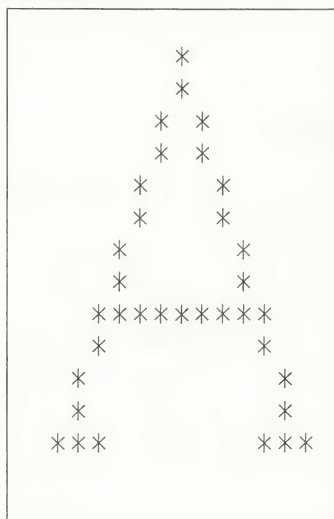
- 1) 出力ポートE 8 H番地にアドレスの下位バイトを出力します。  
出力ポートE 9 H番地にアドレスの上位バイトを出力します。
- 2) E A H番地に何らかのデータ(データ自身は無意味)を出力します。
- 3) 入力ポートE 8 H番地からフォント・データの下位バイトを入力します。  
入力ポートE 9 H番地からフォント・データの上位バイトを入力します。
- 4) E B H番地に何らかのデータ(データ自身は無意味)を出力します。
- 5) 以上を必要な回数くり返します。

E 8 H～E 9 H番地から入力したフォントは16ビット長(8ビット×2組)ですが、このそれぞれのビットが16×16ドットを横に割った1ラインに対応しています。すなわち、16×16ドットのフォントを構成するためには16回のデータ入力を行なうことが必要です。

また、半角文字は8×16ドットで構成されていますが、この場合にはE 9 H番地からの入力データが偶数ラインに、E 8 H番地からの入力データが奇数ラインに対応します。つまり、8×16ドットのフォントを構成するためには8回のデータ入力を行なう事が必要です。

さらに、1/4角文字は8×8ドットで構成されていますが、この場合には4回のデータ入力が必要で、

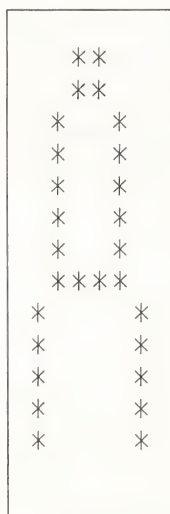
● 16×16ドット (16ワード)



7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0  
(E 9 H) (E 8 H)

0 (E 9 H · E 8 H)  
1 (E 9 H · E 8 H)  
2 (E 9 H · E 8 H)  
3 (E 9 H · E 8 H)  
4 (E 9 H · E 8 H)  
5 (E 9 H · E 8 H)  
6 (E 9 H · E 8 H)  
7 (E 9 H · E 8 H)  
8 (E 9 H · E 8 H)  
9 (E 9 H · E 8 H)  
10 (E 9 H · E 8 H)  
11 (E 9 H · E 8 H)  
12 (E 9 H · E 8 H)  
13 (E 9 H · E 8 H)  
14 (E 9 H · E 8 H)  
15 (E 9 H · E 8 H)

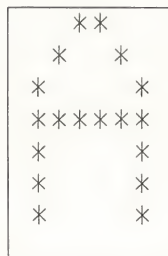
● 8×16ドット (8ワード)



7 6 5 4 3 2 1 0

0 (E 9 H)  
0 (E 8 H)  
1 (E 9 H)  
1 (E 8 H)  
2 (E 9 H)  
2 (E 8 H)  
3 (E 9 H)  
3 (E 8 H)  
4 (E 9 H)  
4 (E 8 H)  
5 (E 9 H)  
5 (E 8 H)  
6 (E 9 H)  
6 (E 8 H)  
7 (E 9 H)  
7 (E 8 H)

● 8×8ドット (4ワード)



7 6 5 4 3 2 1 0

0 (E 9 H)  
0 (E 8 H)  
1 (E 9 H)  
1 (E 8 H)  
2 (E 9 H)  
2 (E 8 H)  
3 (E 9 H)  
3 (E 8 H)



## E 8 H～E 9 H：漢字ROMのアドレスを指定する。

ポ ー ト	E 8 H～E 9 H
I / O	OUTPUT
機 能	漢字ROMのアドレスを指定する。

**解 説** PC-8801に漢字ROMポート（PC-8801-01）を実装した場合の、漢字フォントの読み込み手順を示します。

- 1) 出力ポートE 8 H番地にアドレスの下位バイトを出力します。  
出力ポートE 9 H番地にアドレスの上位バイトを出力します。
- 2) E A H番地に何らかのデータ（データ自身は無意味）を出力します。
- 3) 入力ポートE 8 H番地からフォント・データの下位バイトを入力します。  
入力ポートE 9 H番地からフォント・データの上位バイトを入力します。
- 4) E B H番地に何らかのデータ（データ自身は無意味）を出力します。
- 5) 以上を必要な回数くり返します。

このように、漢字フォントを入力するためには、そのフォントが格納されているアドレスを指定する事が必要です。漢字JISコード(USERS MANUAL等に掲載)と漢字ROMアドレスの間には複雑な関係がありますので、この漢字ROMアドレスの意味を説明します。

漢字ROMには、半角文字、1/4角文字、非漢字、漢字、それぞれのフォント・データが格納されており、ROMアドレスの指定方法もそれぞれ異なります。次ページにそれぞれの場合に分けた、漢字ROMのアドレスと漢字JISコードの関係を示します。また、後述のサンプル・プログラムでは実際に漢字JISコードから漢字フォントのデータを引き出していますので参考にしてください。



●半角文字

ビット

15.....0  
 14.....0  
 13.....0  
 12.....0  
 11.....0  
 10..... J I S 漢字コードの第 7 ビット  
 9 ..... J I S 漢字コードの第 6 ビット  
 8 ..... J I S 漢字コードの第 5 ビット  
 7 ..... J I S 漢字コードの第 4 ビット  
 6 ..... J I S 漢字コードの第 3 ビット  
 5 ..... J I S 漢字コードの第 2 ビット  
 4 ..... J I S 漢字コードの第 1 ビット  
 3 ..... J I S 漢字コードの第 0 ビット  
 2 ]  
 1 ] ... 8 ワードのデータを選択  
 0 ]

●1 / 4 角文字

ビット

15.....0  
 14.....0  
 13.....0  
 12.....0  
 11.....1  
 10.....0  
 9 ..... J I S 漢字コードの第 7 ビット  
 8 ..... J I S 漢字コードの第 6 ビット  
 7 ..... J I S 漢字コードの第 5 ビット  
 6 ..... J I S 漢字コードの第 4 ビット  
 5 ..... J I S 漢字コードの第 3 ビット  
 4 ..... J I S 漢字コードの第 2 ビット  
 3 ..... J I S 漢字コードの第 1 ビット  
 2 ..... J I S 漢字コードの第 0 ビット  
 1 ]  
 0 ] ... 4 ワードのデータを選択

●非漢字

ビット

15.....0  
 14.....0  
 13..... J I S 漢字コードの第 6 ビット  
 12..... J I S 漢字コードの第 5 ビット  
 11..... J I S 漢字コードの第 10 ビット  
 10..... J I S 漢字コードの第 9 ビット  
 9 ..... J I S 漢字コードの第 8 ビット  
 8 ..... J I S 漢字コードの第 4 ビット  
 7 ..... J I S 漢字コードの第 3 ビット  
 6 ..... J I S 漢字コードの第 2 ビット  
 5 ..... J I S 漢字コードの第 1 ビット  
 4 ..... J I S 漢字コードの第 0 ビット  
 3 ]  
 2 ] ... 16 ワードのデータを選択  
 1 ]  
 0 ]

●漢字

ビット

15..... J I S 漢字コードの第 6 ビット  
 14..... J I S 漢字コードの第 5 ビット  
 13..... J I S 漢字コードの第 12 ビット  
 12..... J I S 漢字コードの第 11 ビット  
 11..... J I S 漢字コードの第 10 ビット  
 10..... J I S 漢字コードの第 9 ビット  
 9 ..... J I S 漢字コードの第 8 ビット  
 8 ..... J I S 漢字コードの第 4 ビット  
 7 ..... J I S 漢字コードの第 3 ビット  
 6 ..... J I S 漢字コードの第 2 ビット  
 5 ..... J I S 漢字コードの第 1 ビット  
 4 ..... J I S 漢字コードの第 0 ビット  
 3 ]  
 2 ] ... 16 ワードのデータを選択  
 1 ]  
 0 ]

## E A H～E B H：漢字ROMの読み出しを宣言する。

ポ ー ト	E A H～E B H
-------	-------------

I / O	OUTPUT
-------	--------

機 能	漢字ROMの読み出しを宣言する。
-----	------------------

解 説	PC-8801に漢字ROMポート(PC-8801-01)を実装した場合の、漢字フォントの読み込み手順を示します。
-----	--

- 1) 出力ポートE 8 H番地にアドレスの下位バイトを出力します。  
出力ポートE 9 H番地にアドレスの上位バイトを出力します。
- 2) E A H番地に何らかのデータ(データ自身は無意味)を出力します。
- 3) 入力ポートE 8 H番地からフォント・データの下位バイトを入力します。  
入力ポートE 9 H番地からフォント・データの上位バイトを入力します。
- 4) E B H番地に何らかのデータ(データ自身は無意味)を出力します。
- 5) 以上を必要な回数くり返します。

このように、E A H番地へ何らかのデータを出力する事によって漢字ROMからのデータ読み込み開始を宣言し、データを読み込んだ後に、終了宣言としてE B H番地へ何らかのデータを出力します。

これらの出力は、出力する事自体に意味があり、出力データ自身は何の意味も持ちません。

## サンプル

```

1000 /
1010 / access kanji rom and display font
1020 /
1030 DEFNSG A-Z
1040 /
1050 DIM BIT(15)
1060 FOR I=0 TO 15
1070     BIT(I)=2^I
1080 NEXT
1090 /
1100 WIDTH 40,25:CONSOLE 0,25
1110 /
1120 *LOOP
1130 PRINT CHR$(12);
1140 INPUT 'kanji jis code ';JISCOD$:IF JISCOD$='end' THEN END
1150 JISCOD$=VAL('&h'+JISCOD$)
1160 IF &H0<=JISCOD$ AND JISCOD$<= &HFF THEN GOSUB *HALF :GOTO *LABEL
1170 IF &H100<=JISCOD$ AND JISCOD$<= &H1FF THEN GOSUB *QUARTER :GOTO *LABEL
1180 IF &H2120<=JISCOD$ AND JISCOD$<=&H277F THEN GOSUB *HIKANJI :GOTO *LABEL
1190 IF &H3020<=JISCOD$ AND JISCOD$<=&H4F5F THEN GOSUB *YESKANJI:GOTO *LABEL
1200 GOTO *LOOP
1210 /
1220 *LABEL
1230 IF INKEY$='' THEN 1230
1240 GOTO *LOOP
1250 /
1260 / subroutine for half font
1270 /
1280 *HALF
1290 ROMADDR = JISCOD$ * BIT(3)
1300 /
1310 LOCATE 0,1:PRINT 'kanji rom addr : ';HEX$(ROMADDR)
1320 HI=ROMADDR/256
1330 LO=ROMADDR MOD 256
1340 LOCATE 0,3
1350 /
1360 FOR Y=0 TO 7
1370     OUT &HE8,LO+Y
1380     OUT &HE9,HI
1390     OUT &HEA,&HFF
1400 /
1410     PRINT RIGHT$('000'+HEX$(ROMADDR+Y),4);': ';
1420     I=INP(&HE9)
1430     FOR X=7 TO 0 STEP -1
1440         IF I AND BIT(X) THEN PRINT '*'; ELSE PRINT '.';
1450     NEXT
1460     PRINT
1470 /
1480     PRINT ' ';
1490     I=INP(&HE8)
1500     FOR X=7 TO 0 STEP -1
1510         IF I AND BIT(X) THEN PRINT '*'; ELSE PRINT '.';
1520     NEXT
1530     PRINT
1540 /
1550     OUT &HEB,&HFF
1560 NEXT Y
1570 RETURN
1580 /
1590 / subroutine for quarter font
1600 /
1610 *QUARTER
1620 ROMADDR = &H800 + (JISCOD$ AND &HFF) * BIT(2)
1630 /
1640 LOCATE 0,1:PRINT 'kanji rom addr : ';HEX$(ROMADDR)
1650 HI=ROMADDR/256
1660 LO=ROMADDR MOD 256
1670 LOCATE 0,3
1680 /
1690 FOR Y=0 TO 3
1700     OUT &HE8,LO+Y
1710     OUT &HE9,HI

```

```

1720 OUT &HEA,&HFF
1730 /
1740 PRINT RIGHT$("000"+HEX$(ROMADDR+Y),4);": ";
1750 I=INP(&HE9)
1760 FOR X=7 TO 0 STEP -1
1770 IF I AND BIT(X) THEN PRINT "*"; ELSE PRINT ".";
1780 NEXT
1790 PRINT
1800 /
1810 PRINT " ";
1820 I=INP(&HE8)
1830 FOR X=7 TO 0 STEP -1
1840 IF I AND BIT(X) THEN PRINT "*"; ELSE PRINT ".";
1850 NEXT
1860 PRINT
1870 /
1880 OUT &HEB,&HFF
1890 NEXT Y
1900 RETURN
1910 /
1920 ' subroutine for hi-kanji font
1930 /
1940 *HIKANJI
1950 KU = (JISCODE AND &H700) ¥ &H100
1960 TEN = JISCODE AND &H7F
1970 ROMADDR = 0
1980 ROMADDR = ROMADDR + (TEN ¥ BIT(5)) * BIT(12)
1990 ROMADDR = ROMADDR + (KU * BIT(9))
2000 ROMADDR = ROMADDR + (TEN MOD BIT(5)) * BIT(4)
2010 GOTO *DISPLAY
2020 /
2030 ' subroutine for kanji font
2040 /
2050 *YESKANJI
2060 KU = (JISCODE AND &H1F00) ¥ &H100
2070 TEN = JISCODE AND &H7F
2080 ROMADDR = (TEN ¥ BIT(5)) * BIT(14)
2090 ROMADDR = ROMADDR + (KU * BIT(9))
2100 ROMADDR = ROMADDR + (TEN MOD BIT(5)) * BIT(4)
2110 /
2120 *DISPLAY
2130 LOCATE 0,1:PRINT "kanji rom addr : ";HEX$(ROMADDR)
2140 HI=INT(ROMADDR/&H100)
2150 LO=ROMADDR-INT(ROMADDR/&H100)*&H100
2160 /
2170 LOCATE 0,3
2180 FOR Y=0 TO 15
2190 OUT &HE8,LO+Y
2200 OUT &HE9,HI
2210 OUT &HEA,&HFF
2220 /
2230 PRINT RIGHT$("000"+HEX$(ROMADDR+Y),4);": ";
2240 I=INP(&HE9)
2250 FOR X=7 TO 0 STEP -1
2260 IF I AND BIT(X) THEN PRINT "*"; ELSE PRINT ".";
2270 NEXT
2280 /
2290 I=INP(&HE8)
2300 FOR X=7 TO 0 STEP -1
2310 IF I AND BIT(X) THEN PRINT "*"; ELSE PRINT ".";
2320 NEXT
2330 PRINT
2340 /
2350 OUT &HEB,&HFF
2360 NEXT Y
2370 RETURN

```

kanji jis code ? 217a  
kanji rom addr : 33A0

33A0: .....  
33A1: .....  
33A2: .....  
33A3: .....  
33A4: .....  
33A5: .....  
33A6: .....  
33A7: .....  
33A8: .....  
33A9: .....  
33AA: .....  
33AB: .....  
33AC: .....  
33AD: .....  
33AE: .....  
33AF: .....

kanji jis code ? 21  
kanji rom addr : 108

0108: .....  
.....  
0109: .....  
.....  
010A: .....  
.....  
010B: .....  
.....  
010C: .....  
.....  
010D: .....  
.....  
010E: .....  
.....  
010F: .....  
.....

kanji jis code ? 101  
kanji rom addr : 804

0804: .....  
.....  
0805: .....  
.....  
0806: .....  
.....  
0807: .....  
.....

kanji jis code ? 3021  
kanji rom addr : 6010

6010: .....  
6011: .....  
6012: .....  
6013: .....  
6014: .....  
6015: .....  
6016: .....  
6017: .....  
6018: .....  
6019: .....  
601A: .....  
601B: .....  
601C: .....  
601D: .....  
601E: .....  
601F: .....

kanji jis code ? 23  
kanji rom addr : 118

0118: .....  
.....  
0119: .....  
.....  
011A: .....  
.....  
011B: .....  
.....  
011C: .....  
.....  
011D: .....  
.....  
011E: .....  
.....  
011F: .....  
.....

kanji jis code ? 11c  
kanji rom addr : 870

0870: .....  
.....  
0871: .....  
.....  
0872: .....  
.....  
0873: .....  
.....

kanji jis code ? 4d25  
kanji rom addr : 5A50

5A50: .....  
5A51: .....  
5A52: .....  
5A53: .....  
5A54: .....  
5A55: .....  
5A56: .....  
5A57: .....  
5A58: .....  
5A59: .....  
5A5A: .....  
5A5B: .....  
5A5C: .....  
5A5D: .....  
5A5E: .....  
5A5F: .....

kanji jis code ? 91  
kanji rom addr : 488

0488: .....  
.....  
0489: .....  
.....  
048A: .....  
.....  
048B: .....  
.....  
048C: .....  
.....  
048D: .....  
.....  
048E: .....  
.....  
048F: .....  
.....

kanji jis code ? 1b1  
kanji rom addr : AC4

0AC4: .....  
.....  
0AC5: .....  
.....  
0AC6: .....  
.....  
0AC7: .....  
.....

## FCH:PC-8031からデータを入力する.

ポ ー ト	FCH
-------	-----

I / O	INPUT
-------	-------

機 能	PC-8031からデータを入力する.
-----	--------------------

**解 説** PC-8801では、インテリジェント・タイプのミニ・フロッピー・ディスク・ユニット(PC-8031またはPC-8031-2W)に対するハンドシェイクを行なうために $\mu$ PD8255をモード0で使用しており、ポートA(PA)を、データの入力ポートとして利用します.

この、ポートA(PA)からのデータ入力を行なうのがFCH番地の入力ポートです.

以下に、1バイトのデータを受信する一般的な手順、およびN<sub>88</sub>-BASICによる受信サブルーチンの例を示します.

- 1) RFDをセットし受信する準備ができた事を示します.
- 2) DAVがセットされデータが有効になるまで待ちます.
- 3) データを受信します.
- 4) DACをセットしてデータを受信した事を示します.
- 5) DAVがリセットされるまで待ちます.
- 6) DACをリセットして終了します.

### サンプル

```
100 /
110 / receive a data from pc-8031 subroutine
120 /
130 PA=&HFC:PB=PA+1:PC=PA+2:CW=PA+3
140 /
150 OUT CW,&HB / pc=xx1xxxxx / set rfd (bit5)
160 IF (INP(PC) AND &H1)=0 THEN 160 / pc=xxxxxxxx1 / wait for dav (bit0) set
170 OUT CW,&HA / pc=xx0xxxxx / reset rfd (bit5)
180 A=INP(PA) / / receive data from pa
190 OUT CW,&HD / pc=x1xxxxxx / set dac (bit6)
200 IF INP(PC) AND &H1 THEN 200 / pc=xxxxxxxx0 / wait for dav (bit0) reset
210 OUT CW,&HC / pc=x0xxxxxx / reset dac (bit6)
220 RETURN
230 /
```

## FDH:PC-8031ヘデータを出力する.

ポ ー ト	FDH
I / O	OUTPUT
機 能	PC-8031ヘデータを出力する.

**解 説** インテリジェント・タイプのミニ・フロッピ・ディスク・ユニットに対するハンドシェイクのために、 $\mu$ PD8255のポートB(PB)ヘデータを出力するための出力ポートがFDH番地です.

以下に、1バイトのコマンドおよびデータを送信する場合の一般的な手順とN 88-BASICによる送信サブルーチンの例を示します.

- 1) コマンドの場合のみATNをセットして宣言をします.
- 2) RFDがセットされ受信する準備ができるまで待ちます.
- 3) 次の送信に備えてATNをリセットしておきます.
- 4) コマンドまたはデータを送信します.
- 5) DAVをセットしてデータが有効である事を示します.
- 6) DACがセットされるまで待ちます.
- 7) DAVをリセットします.
- 8) DACがリセットされるまで待ちます.

### サンプル

```

100 /
110 / send a command to pc-8031 subroutine
120 /
130 PA=&HFC:PB=PA+1:PC=PA+2:CW=PA+3
140 /
150 OUT CW,&HF ' pc=1xxxxxxx ' set atn (bit7)
160 /
170 / send a data to pc-8031 subroutine
180 /
190 PA=&HFC:PB=PA+1:PC=PA+2:CW=PA+3
200 /
210 IF (INP(PC) AND &H2)=0 THEN 210 ' pc=xxxxxx1x ' wait for rfd (bit1) set
220 OUT CW,&HE ' pc=0xxxxxxx ' reset atn (bit7)
230 OUT PB,A ' send command or data to pb
240 OUT CW,&H9 ' pc=xxx1xxxx ' set dav (bit4)
250 IF (INP(PC) AND &H4)=0 THEN 250 ' pc=xxxxxx1xx ' wait for dac (bit2) set
260 OUT CW,8 ' pc=xxx0xxxx ' reset dav (bit4)
270 IF INP(PC) AND &H4 THEN 270 ' pc=xxxxxx0xx ' wait for dac (bit2) reset
280 RETURN
290 /

```



## FEH: PC-8031 から制御信号を入力する.

ポ ー ト	FEH
I / O	INPUT
機 能	PC-8031 から制御信号を入力する.

**解 説** PC-8801には、インテリジェント・タイプのミニ・フロッピー・ディスク・ユニット(PC-8031またはPC-8031-2W)に対するインタフェースが内蔵されています。インタフェースには $\mu$ PD8255を使用しており、ディスク・ユニット内の $\mu$ PD8255と3線ハンドシェイク方式で接続されています。

PC-8801に内蔵された $\mu$ PD8255のポートA(PA)がPC-8031からのデータ入力に、ポートB(PB)がPC-8031へのデータ出力に使用され、ポートC(PC)の上位4ビットを制御信号の出力用に下位4ビットを制御信号の入力用に使用します。つまり、 $\mu$ PD8255をモード0で使用している事になり、入力ポートFEH番地は、ポートC(PC)からの制御信号を入力するためのポートです。

以下に、入力した制御信号の各ビットの意味を示します。

第2ビット(DAC).....DATA ACCEPTED

第1ビット(RFD).....READY FOR DATA

第0ビット(DAV).....DATA VALID

## FFH:PC-8031へ制御信号を出力する。

ポ ー ト	FFH
I / O	OUTPUT
機 能	PC-8031へ制御信号を出力する。

**解 説** インテリジェント・タイプのミニ・フロッピ・ディスク・ユニットに対するハンドシェイクのために、 $\mu$ PD8255のポートC(PC)上位4ビットへ制御信号を出力するための出力ポートがFFH番地です。

以下にポートC(PC)上位4ビットの各ビットの意味を示します。

第7ビット(ATN).....ATTENTION

第6ビット(DAC).....DATA ACCEPTED

第5ビット(RFD).....READY FOR DATA

第4ビット(DAV).....DATA VALID

しかし、FFH番地に出力したデータがそのままポートC(PC)にセットされるわけではありません( $\mu$ PD-8255の仕様)。次に、実際にFFH番地に出力するデータの各ビットの意味を示します。

第7ビット.....出力ポートFFH番地はポートC(PC)のセット以外にも、 $\mu$ PD8255のモード設定を行なう場合にも使用します。この第7ビットを1に指定した場合にはモード設定を選択し、0を指定した場合にはポートC(PC)に対するビットのセット、リセットを選択します。ちなみに $\mu$ PD8255のイニシアライズはFFH番地に91Hを出力する事で行ないます。

第3ビット～第1ビット.....この3ビットによって、ポートC(PC)の8ビットの中のどのビットに対して、セット、リセットを行なうかを指定します。

第0ビット.....第3ビット～第1ビットで指定したビットをセットするかリセットするのかを指定するビットです。

たとえば、ポートCの第7ビット(PC7)を1にセットしてATN(ATTENTION)を宣言したければ、FFH番地に0FHを出力すれば良い事になります。



## ***SYSTEM WORKING AREA***



E 6 0 0 H 単精度減算用サブルーチン。  
 E 6 0 E H 乱数用。乱数 =  $F$  (前の乱数  $\times X + Y$ ) の  $F$  に関する値 (0 0 H ~ A B H)。  
 E 6 0 F H 乱数用。下位 2 ビットで 3 種類の  $Y$  を選択。  
 E 6 1 0 H 乱数用。下位 3 ビットで 8 種類の  $X$  を選択。  
 E 6 1 1 H 単精度実数型の乱数用定数 8 種類。  
 E 6 3 1 H 前に発生した乱数の退避用。  
 E 6 3 5 H USR 関数, USR 0 ~ USR 9 のコール先アドレス。  
 E 6 4 9 H エラー・コード (ERR)。  
 E 6 4 A H 未使用。  
 E 6 4 B H プリンタのヘッド位置。  
 E 6 4 C H 出力用フラグ。0 0 H で CRT を 0 1 H でプリンタを選択。  
 E 6 4 D H プリンタの横幅。  
 E 6 4 E H プリンタの桁数。WIDTH LPRINT で指定した値。  
 E 6 4 F H CRT スクリーンの桁数。WIDTH で指定した値。  
 E 6 5 0 H CRT スクリーンの横幅。  
 E 6 5 1 H 未使用。  
 E 6 5 2 H CRT 出力フラグ。  
 E 6 5 3 H 機械語プログラムのロード、セーブ用フラグ。  
 E 6 5 4 H フリー・エリアの上限。  
 E 6 5 6 H 実行中の行番号。ダイレクト・モード時は F F F F H。  
 E 6 5 8 H テキストのトップ・アドレス。  
 E 6 5 A H “/ 0” または “OV” のためのエラー・メッセージ・ポインタ。  
 E 6 5 C H ESC ! で出力する ID コード (NEC 0 0 0 0 0 0 0 1  $C_R L_F$ )。  
 E 6 6 9 H 内蔵マシン語モニタのジャンプ・テーブル。  
 E 6 8 4 H 未使用。  
 E 6 9 C H テキストの構文解析用 (1 E H, 1 0 H)。  
 E 6 9 E H ラスト・ドライブ ID。  
 E 6 9 F H ターミナル・モード・フラグ。  
 E 6 A 0 H ターミナル・モード時に BASIC の出力制御で使用。  
 E 6 A 1 H ターミナル・モード時に BASIC で入力したキャラクタ・コード。  
 E 6 A 2 H ターミナル・モードのプリンタ・イネーブル・フラグ。  
 E 6 A 3 H 1 1 2 A H 番地から使用。  
 E 6 A 4 H CMT ヘッダ・サーチ・フラグ。  
 E 6 A 5 H ターミナル・モード。

- E 6 A 6 H グラフィックでの縦の最大表示ドット数. 0で200, 1で400ドット.
- E 6 A 7 H カーソル表示フラグ.
- E 6 A 8 H カーソル表示のためにCRTコントローラに与えるコマンド.
- E 6 A 9 H CMT出力フラグ.
- E 6 A A H 未使用.
- E 6 A B H 行番号. ピリオドで指定する行番号の値.
- E 6 A D H エディタでインサート・モード用のフラグ.
- E 6 A E H ライト・ペンの補正值 (0 6 H).
- E 6 A F H ライト・ペンの補正值 (3 6 H).
- E 6 B 0 H 実際のスクロール開始ライン (1~25).
- E 6 B 1 H 実際のスクロール範囲 (1~25).
- E 6 B 2 H CONSOLEで指定したスクロール開始ライン (1~25).
- E 6 B 3 H CONSOLEで指定したスクロール範囲 (1~25).
- E 6 B 4 H 現在のアトリビュート・コード.
- E 6 B 5 H ヌル・キャラクタ・コード.
- E 6 B 6 H 0 0 H以外の時CRTにコントロール・コードを表示する.
- E 6 B 7 H 未使用.
- E 6 B 8 H 0 0 H以外の時ファンクション・キー表示を行なう.
- E 6 B 9 H カラー／白黒フラグ. FFHでカラー・モード, 0 0 Hで白黒モード.
- E 6 B A H COPY用シフト・モード.
- E 6 B B H EDITでROLL-UPのための行番号.
- E 6 B D H EDITでROLL-DOWNのための行番号.
- E 6 B F H USARTエラー.
- E 6 C 0 H 出力ポート30H番地への出力データ.
- E 6 C 1 H 出力ポート40H番地への出力データ.
- E 6 C 2 H 出力ポート31H番地への出力データ.
- E 6 C 3 H 出力ポートE4H番地への出力データ (インタラプト・レベル).
- E 6 C 4 H VRAMのスタート・アドレス.
- E 6 C 6 H 時間制限したキーボードからの入力用フラグ.
- E 6 C 7 H TIME \$によるインタラプト指定フラグ (FFHで指定).
- E 6 C 8 H 未使用.
- E 6 C 9 H ライン・バッファのオーバーフロー・フラグ.
- E 6 C A H コントロール・コード入力用.
- E 6 C B H インタラプト・バッファのためのテーブルのトップ・アドレス.



E 6 C D H キー入力フラグ。0 0 Hの時のみキー入力を受けつける。  
 E 6 C E H ファンクション・キー・フラグ。  
 E 6 C F H オート・リピート用カウンタ。  
 E 6 D 0 H キー・スキャン用ポートからの入力データ1 2 バイト。  
 E 6 E 8 H USARTフラグ。CMTまたはS I O。  
 E 6 E 9 H COPY用カウンタ。  
 E 6 E B H L P T：用オープン・フラグ（F F Hでオープン）。  
 E 6 E C H COM1：用W I D T H。  
 E 6 E D H USARTフラグ。  
 E 6 E E H インタラプト・テーブルの数。  
 E 6 E F H インタラプトによるG O S U Bのテーブル・アドレス（E E C B H）。  
 E 6 F 1 H インタラプト・フラグ。  
 E 6 F 2 H プログラマブル・ファンクション・キー（f・1）のバッファ。  
 E 7 0 2 H プログラマブル・ファンクション・キー（f・2）のバッファ。  
 E 7 1 2 H プログラマブル・ファンクション・キー（f・3）のバッファ。  
 E 7 2 2 H プログラマブル・ファンクション・キー（f・4）のバッファ。  
 E 7 3 2 H プログラマブル・ファンクション・キー（f・5）のバッファ。  
 E 7 4 2 H プログラマブル・ファンクション・キー（f・6）のバッファ。  
 E 7 5 2 H プログラマブル・ファンクション・キー（f・7）のバッファ。  
 E 7 6 2 H プログラマブル・ファンクション・キー（f・8）のバッファ。  
 E 7 7 2 H プログラマブル・ファンクション・キー（f・9）のバッファ。  
 E 7 8 2 H プログラマブル・ファンクション・キー（f・1 0）のバッファ。  
 E 7 9 2 H ターミナル・モード時のファンクション・キー（f・6）のバッファ。  
 E 7 A 2 H ターミナル・モード時のファンクション・キー（f・7）のバッファ。  
 E 7 B 2 H ターミナル・モード時のファンクション・キー（f・8）のバッファ。  
 E 7 C 2 H ターミナル・モード時のファンクション・キー（f・9）のバッファ。  
 E 7 D 2 H ターミナル・モード時のファンクション・キー（f・1 0）のバッファ。  
 E 7 E 2 H L I N E用のジャンプ・テーブル1。  
 E 7 E 5 H L I N E用のジャンプ・テーブル2。  
 E 7 E 8 H ワーク・エリアを除いたRAMの上限（E 5 F F H）。  
 E 7 E A H インタラプト処理ルーチン。  
 E 8 2 6 H 内蔵モニタ用サブルーチン。  
 E 8 4 0 H 変数名バッファ。  
 E 8 7 7 H 変数ルーチン、ワーク・エリア。

E 8 7 9 H インターメディアイト・コード (中間言語) バッファ。  
 E 9 B 9 H インプット・バッファ 2 5 6 バイト。  
 E A B 9 H 未使用。  
 E A B B H C H A I N 用。  
 E A B C H 配列宣言フラグ。  
 E A B D H フローティング・アキュムレータの型 (0, 2, 4 または 8)。  
 E A B E H 中間コード, リスト・フラグ 1。  
 E A B F H 中間コード, リスト・フラグ 2。  
 E A C 0 H R S T 1 0 H 用ポインタ。  
 E A C 2 H R S T 1 0 H 用データ。  
 E A C 3 H R S T 1 0 H 用でデータの型 (0, 2, 4 または 8)。  
 E A C 4 H R S T 1 0 H 用データの値。  
 E A C C H ストリング・エリアの上限 (スタックの下限)。  
 E A C E H 文字式にカッコを使用する際のスタック・ポインタ。  
 E A D 0 H 文字式にカッコを使用する際のストリング・ディスクリプタ 1 1 組。  
 F A F 1 H ストリング・エリアの下限。  
 F A F 3 H 式処理ルーチン, ワーク・エリア。  
 E A F 5 H ガベージ・コレクション用ワーク・エリア。  
 E A F 7 H F O R 用ワーク・エリア。  
 E A F 9 H D A T A 用エラーの行番号。  
 E A F B H 変数のタイプ, 0 で普通, 6 4 で F O R, 8 0 で配列。  
 E A F C H R E A D, I N P U T 用フラグ。  
 E A F D H 変数アドレス。  
 E A F F H 行番号, 行アドレス・フラグ, 0 0 H で行番号, 0 D H でアドレスに変換。  
 E B 0 0 H A U T O 実行中フラグ。  
 E B 0 1 H A U T O 行番号。  
 E B 0 3 H A U T O 増分。  
 E B 0 5 H 実行アドレス。  
 E B 0 7 H 実行スタック。  
 E B 0 9 H エラー発生行番号 (E R L)。  
 E B 0 B H エラー発生アドレス。  
 E B 0 D H エラー・トラップ先のアドレス (O N E R R O R G O T O で指定)。  
 E B 0 F H エラー回数。  
 E B 1 0 H ワーク・エリア。

EB12H ブレークした行番号。  
EB14H CONTを行なうための中断アドレス（0でCONT不可能）。  
EB16H ファイル・バッファの上限（ラベル・エリアの下限）。  
EB18H テキストのエンド・アドレス。  
EB1AH ラベル定義済フラグ。  
EB1BH ラベル領域の上限（単純変数領域の下限）。  
EB1DH 単純変数領域の上限（配列変数領域の下限）。  
EB1FH 配列変数領域の上限（PAINT用ワーク・エリアの下限）。  
EB21H 現在読み込んでいるDATAのアドレス。  
EB23H 型宣言を行なった変数の型。A～Zの26バイト。  
EB3DH FN用引数テーブルのポインタ。  
EB3FH FN用引数テーブルのバイト数。  
EB41H FN用引数テーブル（型，名称，処理アドレス）。  
EBA5H FN用引数テーブルのポインタ。  
EBA7H FN用引数テーブルのバイト数。  
EBA9H FN用引数テーブル（型，名称，処理アドレス）。  
EC0DH 変数，引数フラグ。  
EC0EH 変数または引数のサーチ終了アドレス。  
EC10H 変数または引数のサーチ終了アドレス。  
EC12H カベージ・コレクション用ワーク・エリア。  
EC13H FNのレベル。  
EC15H INPUT用。  
EC16H NEXT用アドレス。  
EC18H NEXT用フラグ。  
EC19H FOR用データ。  
EC1DH FOR用行番号。  
EC1FH OPTION BASEで指定した値（0または1）。  
EC20H OPTION BASE実行フラグ（0で未実行，1または2で実行済）。  
EC21H LIST出力時の空白出力フラグ。  
EC22H INPUT用の疑問符表示フラグ。  
EC23H スtring・エリアの退避用。  
EC25H 未使用。  
EC27H SAVE用のプロテクト・フラグ。  
EC28H LOAD用のパス・ワード・フラグ。

EC29H 0以外の場合, LIST, LLIST, POKE, MONが実行不可能.  
EC2AH CHAIN用MERGEフラグ.  
EC2BH CHAIN用DELETEフラグ.  
EC2CH CHAIN用DELETE開始行番号.  
EC2EH CHAIN用DELETE終了行番号.  
EC30H CHAIN用フラグ.  
EC31H CHAIN用実行行番号.  
EC33H SWAP用変数退避バッファ.  
EC3BH TRON実行フラグ (00HでTROFF, AFHでTRON).  
EC3CH フローティング・アキュムレータ (FACC) の無効桁.  
EC3DH フローティング・アキュムレータ (FACC).  
EC45H フローティング・アキュムレータ (FACC) の符号保持用.  
EC46H オーバーフロー (OV) 表示フラグ.  
EC47H オーバーフロー (OV) 表示フラグの退避用.  
EC48H 10進化フラグ.  
EC49H サブ・フローティング・アキュムレータの無効桁.  
EC4AH サブ・フローティング・アキュムレータ.  
EC52H 数値データからキャラクタ・コードへの変換バッファ.  
EC6DH 倍精度除算用レジスタ.  
EC75H 倍精度乗算用レジスタ.  
EC7DH 最大ドライブ数.  
EC7EH 最大ファイル数.  
EC7FH ファイル・アドレス・テーブルのアドレス.  
EC81H ドライブ・アドレス・テーブルのアドレス.  
EC83H ファイル#0のデータ部のアドレス (VARPTRで得る値+9).  
EC85H ドライブ・ナンバ.  
EC86H ドライブ・アドレス.  
EC88H ファイル・アドレス.  
EC8AH ファイル名アドレス.  
EC8CH ファイル名カウンタ.  
EC8DH ファイル名セクタ.  
EC8EH LOADでRオプション・フラグ.  
EC8FH ファイル名バッファ1.  
EC88H ファイル名バッファ2.

ECA1H 未使用。  
ECA3H ターミナル・モード用実行フラグ。  
ECA4H SAVE用フラグ。  
ECA5H DSKF用の最大トラック番号（片面あたりのトラック数－1）。  
ECA6H DSKF用の1トラックあたりのセクタ数。  
ECA7H DSKF用の片面、両面フラグ（0で片面，1で両面）。  
ECA8H DSKF用の1トラックあたりのクラスタ数。  
ECA9H DSKF用のボリュームあたりのクラスタ数。  
ECAAH DSKF用のディレクトリ・トラック番号。  
ECABH DSKF用の1クラスタあたりのセクタ数。  
ECACH DSKF用のFATの開始セクタ番号。  
ECADH DSKF用のFATの終了セクタ番号。  
ECAEH DSKF用のFATのセクタ数。  
ECAFH DSKF用のディスク属性の入っているセクタ番号。  
ECB0H 未使用。  
ECB1H BSAVE用の終了アドレス。  
ECB3H 未使用。  
ECB4H ディスク・アクセス時のエラー・カウンタ。  
ECB5H 未使用。  
ECB6H ディスクのエラー・フラグ。  
ECB7H 未使用。  
ECBBH ジャンプ・テーブル。  
EECBH ONインタラプトGOSUB用のテーブル。  
EF0AH サブROMコール時のアキュムレータ退避用。  
EF0BH サブROMコール時のレジスタHL退避用。  
EF0DH COPY用フラグ。  
EF0EH 出力ポートE6H番地への出力データ。  
EF0FH 出力ポートF4H番地（F8H番地）への出力データ。  
EF10H ディスク用タイム・アウト・カウンタ。  
EF11H ディスク用フラグ。  
EF12H テキスト・ウィンド用のDMA転送アドレス。  
EF14H ディスク用の待時間。  
EF15H ミニ・フロッピ用の倍速転送フラグ。  
EF16H DMAフラグ。

EF17H FDCコマンド。  
EF18H FDCドライブ・サーフェス。  
EF19H FDCトラック。  
EF1AH FDCセクタ。  
EF1BH 転送セクタ数。  
EF1CH DMAアドレス。  
EF1EH ディスク用600進カウンタ。  
EF20H ディスク待時間。  
EF21H DMAコントローラ用コマンド・バイト。  
EF22H FDC用コマンド・バイト。  
EF23H FDCのステータス。  
EF24H FDC用リトライ・カウンタ。  
EF25H 未使用。  
EF26H FDCの結果・ステータス(ST0)。  
EF27H FDCの結果・ステータス(ST1)。  
EF28H FDCの結果・ステータス(ST2)。  
EF29H FDCの結果・ステータス(C)。  
EF2AH FDCの結果・ステータス(H)。  
EF2BH FDCの結果・ステータス(R)。  
EF2CH FDCの結果・ステータス(N)。  
EF2DH ドライブ・ステータス。  
EF3DH エラー・ステータス。  
EF3EH DMAフラグ。  
EF3FH インタラプト・レベルの退避用。  
EF40H レザルト・ステータス。  
EF41H ディスクのマージン・ポイント(2バイトが4組)。  
EF49H BLOAD時のRオプション・フラグ(00Hまたは52H)。  
EF4AH ディスクのセクタ数。  
EF4BH ドライブ・ステータスのアドレス(ミニでEF2DH, 標準でEF35H)。  
EF4DH マージン・ポートのアドレス(ミニでF9H, 標準でF5H)。  
EF4EH FDコントロール・アドレス(ミニでF8H, 標準でF4H)。  
EF4FH FDコントロール・アドレス(ミニでF8H, 標準でF4H)。  
EF50H 最大トラック・ナンバ(ミニで27H, 標準で4CH)。  
EF51H リード・タイム(ミニで14H, 標準で26H)。



EF52H EOT (ミニで10H, 標準で1AH).  
 EF53H ミニで33H, 標準で36H.  
 EF54H FDCのステータス (ミニでFAH, 標準でF6H).  
 EF55H FDCのデータ (ミニでFBH, 標準でF7H).  
 EF56H 出力ポートF3H番地への出力データ (ミニで10H, 標準で20H).  
 EF57H SPECIFYデータ (ミニで3AH, 標準でBAH).  
 EF58H SPECIFYデータ (ミニで18H, 標準で42H).  
 EF59H DMACのコマンド (ミニでA5H, 標準でA6H).  
 EF5AH DMACのチャンネル・アドレス (ミニで60H, 標準で62H).  
 EF5BH DMACのアドレス (ミニで60H, 標準で60H).  
 EF5CH 出力ポートF3H番地への出力データ (ミニで01H, 標準で02H).  
 EF5DH ドライブID (0~3).  
 EF5EH 未使用.  
 EF5FH ドライブ・ナンバ.  
 EF60H 接続されている標準DMA転送方式のドライブ数.  
 EF61H 接続されているミニDMA転送方式のドライブ数.  
 EF62H 接続されているインテリジェント・タイプのドライブ数.  
 EF63H インテリジェント・タイプのドライブが片面の時00H, 両面の時80H.  
 EF64H ドライブID表 (0で標準DMA, 1でミニDMA, 2で片面, 3で両面).  
 EF70H 未使用.  
 EF71H ターミナル・モード送出スイッチ.  
 EF72H ターミナル・モード時の全二重, 半二重スイッチ.  
 EF73H ターミナル・モード時のレジスタHL退避用.  
 EF75H ターミナル・モード時のコントロール・コード表示スイッチ.  
 EF76H ターミナル・モード時のストリング・エリア退避用.  
 EF78H 多目的に使用.  
 EF7FH 入力ポート30H番地からの入力データ.  
 EF80H 入力ポート31H番地からの入力データ.  
 EF81H ライト・ペンの縦座標.  
 EF82H ライト・ペンの横座標.  
 EF83H エディット時のカーソル縦座標.  
 EF84H エディット時のカーソル横座標.  
 EF85H エディット時のカーソル表示フラグ.  
 EF86H カーソル縦座標 (1~25).



EF87H カーソル横座標 (1~80).  
 EF88H CRTスクリーンの行数 (WIDTHで指定した値).  
 EF89H CRTスクリーンの桁数 (WIDTHで指定した値).  
 EF8AH 制御したアトリビュート・エリアのアドレス.  
 EF8CH 制御したアトリビュート・コード.  
 EF8DH 直前に制御したアトリビュート・エリアの桁.  
 EF8EH 直前に制御したアトリビュート・エリアの次桁.  
 EF8FH 直前に表示したキャラクタのコード.  
 EF90H 00Hのときファンクション・キー・ノーマル, 05Hのときシフト.  
 EF91H COPY用バッファ.  
 EF99H CRTのリンク情報 (00Hのとき前行とリンクしている).  
 EFB3H 未使用.  
 EFB4H 00Hのときノーマル・モード, 01HのときEDITモード.  
 EFB5H 直前に実行したテキストのポインタ.  
 EFB7H 直前にサーチしたテキストのポインタ.  
 EFB9H EDIT用フラグ.  
 EFBAH エラー発生時のバッファのポインタ.  
 EFBCH エラー発生時のポインタ.  
 EFBEH エラー発生時の行番号.  
 EFC0H HELP (CTRL-A) 用のエラー・ポインタ.  
 EFC2H HELP (CTRL-A) 用のカーソル座標.  
 EFC4H キーボードからの入力時に時間制限を行なう長さ.  
 EFC6H キーボードからの入力時に時間制限を行なうためのカウンタ.  
 EFC8H キーボードからの入力時に時間制限を行なうための60進カウンタ.  
 EFC9H ON TIME\$用の1200進カウンタ.  
 EFCAH ON TIME\$用の60進カウンタ.  
 EFCBH ON TIME\$用のカウンタ.  
 EFCDH インタラプト・バッファのテーブル2組.  
 EFD9H キー・スキャン・バッファ.  
 EFF9H キー・スキャン・ロール・オフ.  
 EFFAH キー・スキャン・データ.  
 EFFBH キー・スキャン・ポインタ.  
 EFFDH キー・スキャン・データ.  
 EFFE H キー・スキャンで直前に入力したキャラクタ・コード.

E F F F H キー・スキャンで直前の特殊キーの状態。  
 F 0 0 0 H キー・スキャンで特殊キーの状態。  
 F 0 0 1 H キー・スキャンでCAPSキーの状態 (0 0 Hでオフ, 8 0 Hでオン)。  
 F 0 0 2 H 1でシフト, 2でCTRL, 3でカナ, 4でカナ・シフト, 5でGRPH。  
 F 0 0 3 H ファンクション・キー・ポイント。  
 F 0 0 5 H USARTのファイル・タイプ。  
 F 0 0 6 H CMTのヘッダ・サーチ・フラグ。  
 F 0 0 7 H CMTのロード・フラグ。  
 F 0 0 8 H CMTの入力データ。  
 F 0 0 9 H CMTのボーレート (FAHで6 0 0 ボー, FBHで1 2 0 0 ボー)。  
 F 0 0 A H CMTのベリファイ・フラグ。  
 F 0 0 B H ターミナル・モードXパラメータ・フラグ。  
 F 0 0 C H ターミナル・モードでのデータ転送一時停止フラグ。  
 F 0 0 D H タイマのSECOND (秒:BCDコード)。  
 F 0 0 E H タイマのMINUTE (分:BCDコード)。  
 F 0 0 F H タイマのHOUR (時:BCDコード)。  
 F 0 1 0 H タイマのDAY (日:BCDコード)。  
 F 0 1 1 H タイマのMONTH (月:バイナリ・コード)。  
 F 0 1 2 H タイマのYEAR (年:BCDコード)。  
 F 0 1 3 H PUT@の条件。  
 F 0 1 4 H PUT@でドットをセットする場合のカラー。  
 F 0 1 5 H PUT@でドットをリセットする場合のカラー。  
 F 0 1 6 H PUT@のドット・セット・フラグ。  
 F 0 1 7 H PUT@のドット・リセット・フラグ。  
 F 0 1 8 H PUT@のセット用ビット・パターン。  
 F 0 1 9 H PUT@のリセット用ビット・パターン。  
 F 0 1 A H グラフィック横座標 (0 ~ 6 3 9)。  
 F 0 1 C H グラフィック縦座標 (0 ~ 1 9 9)。  
 F 0 1 E H ドット・グラフィック用カラー (0 ~ 7)。  
 F 0 1 F H ドット・グラフィックでドットをリセットする場合のカラー (0 ~ 7)。  
 F 0 2 0 H ボーダー・カラー (0 ~ 7)。  
 F 0 2 1 H GET@で読み込むグラフィック・パターンの横のドット数。  
 F 0 2 3 H GET@で読み込むグラフィック・パターンの縦のドット数。  
 F 0 2 5 H ラインスタイル。

F 0 2 7 H 直前のグラフィック横座標 ( 0 ~ 6 3 9 ).  
F 0 2 9 H 直前のグラフィック縦座標 ( 0 ~ 1 9 9 ).  
F 0 2 B H ビューポートの左上の頂点の横座標.  
F 0 2 D H ビューポートの左上の頂点の縦座標.  
F 0 2 F H ビューポートの右下の頂点の横座標.  
F 0 3 1 H ビューポートの右下の頂点の縦座標.  
F 0 3 3 H グラフィック座標に対応する G V R A M のアドレス.  
F 0 3 5 H グラフィック座標に対応するビット・パターン.  
F 0 3 6 H グラフィック座標に対応する B L U E 信号の有無 ( F F H または 0 0 H ).  
F 0 3 7 H グラフィック座標に対応する R E D 信号の有無 ( F F H または 0 0 H ).  
F 0 3 8 H グラフィック座標に対応する G R E E N 信号の有無 ( F F H または 0 0 H ).  
F 0 3 9 H ドット・グラフィックの縦の最大値 ( 2 0 0 または 4 0 0 ).  
F 0 3 B H L I N E 用アドレス.  
F 0 3 D H L I N E 用 B L U E 信号の有無 ( F F H または 0 0 H ).  
F 0 3 E H L I N E 用 R E D 信号の有無 ( F F H または 0 0 H ).  
F 0 3 F H L I N E 用 G R E E N 信号の有無 ( F F H または 0 0 H ).  
F 0 4 0 H L I N E 用パターン.  
F 0 4 2 H L I N E 用サブルーチンのアドレス.  
F 0 4 4 H L I N E 用カラー ( 0 ~ 7 ).  
F 0 4 5 H L I N E でドットをリセットする場合のカラー ( 0 ~ 7 ).  
F 0 4 6 H P O I N T 用アドレス.  
F 0 4 8 H P O I N T 用カラー.  
F 0 4 9 H 未使用.  
F 0 4 B H C O P Y モード.  
F 0 4 C H L I N E アドレス.  
F 0 4 E H L I N E カラー.  
F 0 4 F H P A I N T 用タイルの格納されているストリング・エリアの先頭アドレス.  
F 0 5 1 H 未使用.  
F 0 5 2 H P A I N T 用タイルの格納されているストリング・エリアの終了アドレス.  
F 0 5 4 H P A I N T 用タイルのバック・グラウンド・ストリング.  
F 0 5 6 H P A I N T フラグ.  
F 0 5 7 H P A I N T 用タイルのバイト数 ( 0 から ).  
F 0 5 8 H 未使用.  
F 0 5 9 H P A I N T 用タイルのバック・グラウンド・ストリング.

F05CH 不定.  
 F062H PAINT用キューの最大長.  
 F064H PAINT用キューの長さ.  
 F066H PAINT用キューのライト・カウンタ.  
 F068H PAINT用キューのリード・カウンタ.  
 F06AH 不定.  
 F071H PUT@のとき80H, GET@のとき00H.  
 F072H 不定.  
 F087H SCREENで指定した画面モード(0~2).  
 F088H SCREENで指定した画面スイッチ(0~3).  
 F089H SCREENで指定したアクティブ・ページ.  
 F08AH SCREENで指定したアクティブ・ページの数.  
 F08BH SCREENで指定したアクティブ・ページのポート.  
 F08CH SCREENで指定したディスプレイ・ページ(0~7).  
 F08DH SCREENで指定した縦の最大ドット数(200または400).  
 F08FH WINDOWフラグ(01Hのとき実行済, 00Hのとき未実行).  
 F090H ビューポートの横の最大値.  
 F092H ビューポートの縦の最大値.  
 F094H ウィンドウの横の最大値.  
 F098H ウィンドウの縦の最大値.  
 F09CH ビューポートとウィンドウの横方向の比.  
 F0A0H ビューポートとウィンドウの縦方向の比.  
 F0A4H 直前のワールド横座標.  
 F0A8H 直前のワールド縦座標.  
 F0ACH 多目的に使用.  
 F0B0H ウィンドウの左上の頂点の横座標.  
 F0B4H ウィンドウの左上の頂点の縦座標.  
 F0B8H ウィンドウの右下の頂点の横座標.  
 F0BCH ウィンドウの右下の頂点の縦座標.  
 F0C0H VIEW用.  
 F0C2H VIEW用.  
 F0C3H VIEW用.  
 F0C5H VIEW用.  
 F0C6H VIEW用.

F0C8H VIEW用。  
 F0CAH VIEW用。  
 F0CBH VIEW用。  
 F0CCH VIEW用。  
 F0CEH VIEW用。  
 F0D0H 未使用。  
 F0D3H RS-232C用インタラプト・バッファ。  
 F153H RS-232C入力用SI, SOフラグ。  
 F154H RS-232C出力用SI, SOフラグ。  
 F155H モニタからメインROM内をコールするためのサブルーチン。  
 F16CH モニタから裏ROMを選択するサブルーチン。  
 F175H モニタからメインROMを選択するサブルーチン。  
 F184H モニタ用ブロック転送(LDIR)サブルーチン。  
 F187H モニタ用ブロック転送(LDDR)サブルーチン。  
 F18AH モニタでレジスタHLで示すメモリのデータをアキュムレータにロードする。  
 F1B5H モニタからメインROMモードで書き込むサブルーチン。  
 F1BCH モニタ用のエラー・トラップ。  
 F1C2H モニタのGコマンドで、メインROMを選択してジャンプするルーチン。  
 F1C8H モニタのブレータ・ポイント(FFH)トラップ。  
 F1CEH モニタのベース・フラグ(48Hのとき16進, 51Hのとき8進)。  
 F1CFH モニタのブレイク・ポイント・フラグ(0のときブレイク・ポイント無し)。  
 F1D0H ブレイク・ポイントのデータ退避用1。  
 F1D1H ブレイク・ポイントのアドレス1。  
 F1D3H ブレイク・ポイントのデータ退避用2。  
 F1D4H ブレイク・ポイントのアドレス2。  
 F1D6H モニタのDコマンドのアドレス。  
 F1D8H モニタのSまたはEコマンドのアドレス。  
 F1DAH モニタのLコマンドのアドレス。  
 F1DCH モニタで直前のアドレス。  
 F1DEH モニタで直前に入力されたコマンド。  
 F1DFH プリント制御用。  
 F1E0H 数値データ入力フラグ1。  
 F1E1H 0でメインROM, 1で裏ROM, 3でサブROM。  
 F1E2H CMTのファイル・ネーム・バッファ。

F1EFH モニタのベリファイ・フラグ(00Hでリード, FFHでベリファイ).  
 F1F0H 多目的に使用.  
 F1F1H 数値データ入力フラグ2.  
 F1F2H ディスク・ダンプ用.  
 F1F4H プリンタ・フラグ(00Hでディスエーブル, FFHでイネーブル).  
 F1F5H テキスト・ポインタ(レジスタHL) 退避用.  
 F1F7H スタック・ポインタ(SP) 退避用.  
 F1F9H 出力ポート70H番地への出力データ.  
 F1FAH EDC9H番地のデータ退避用.  
 F1FBH EDCAH番地のデータ退避用.  
 F1FCH EDCBH番地のデータ退避用.  
 F1FDH スタック・ポインタ(SP) 退避用.  
 F1FFH プログラム・カウンタ(PC) 退避用.  
 F201H インタラプト・ベクタ・レジスタ(レジスタI) 退避用.  
 F203H レジスタIY退避用.  
 F205H レジスタIX退避用.  
 F207H レジスタHL' 退避用.  
 F209H レジスタDE' 退避用.  
 F20BH レジスタBC' 退避用.  
 F20DH レジスタAF' 退避用.  
 F20FH レジスタHL退避用.  
 F211H レジスタDE退避用.  
 F213H レジスタBC退避用.  
 F215H レジスタAF退避用.  
 F217H 16進から8進への変換バッファ.  
 F21EH 未使用.  
 F300H インタラプト・ベクトルでCMTおよびSIOインタラプト(E7EAH).  
 F302H インタラプト・ベクトルでVRTCインタラプト(E808H).  
 F304H インタラプト・ベクトルでリアルタイム・インタラプト(E80EH).  
 F306H インタラプト・ベクトルでユーザー・インタラプトINT4(E814H).  
 F308H インタラプト・ベクトルでユーザー・インタラプトINT3(E814H).  
 F30AH インタラプト・ベクトルでユーザー・インタラプトINT2(E814H).  
 F30CH インタラプト・ベクトルでFDD1(E81AH).  
 F30EH インタラプト・ベクトルでFDD2(E820H).

F 3 1 0 H    インタラプト・ベクトルで 8 レベル未使用.

F 3 2 0 H    未使用.

F 3 C 8 H    テキスト V R A M.

F F F 8 H    未使用.



# ***PC-8801 CIRCUIT DIAGRAM***



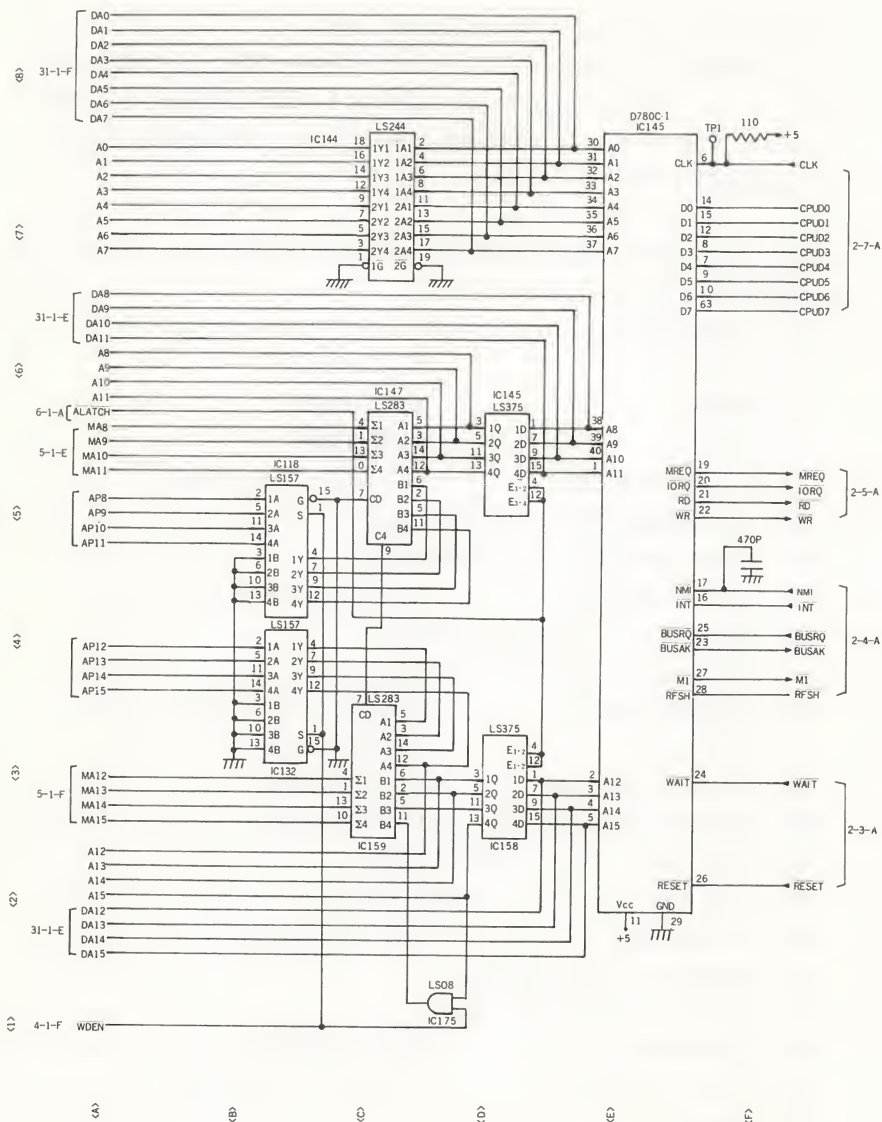
本章では、PC-8801の全回路図を紹介します。

これらの回路図は全て秀和システムトレーディング株式会社において調査したもので、メーカーが保障したものではありません。あくまでも参考図としてお使いください。したがって、本回路図集に関してメーカー等に直接問い合わせることはご遠慮ください。

本回路図集は、PC-8801本体を31のブロックに分けた上で、31ページの回路図としてまとめたものです。

以下に、回路図集中のページングと各ブロックの内容を示します。

- |                              |                 |
|------------------------------|-----------------|
| 1. CPU(1)                    | 2. CPU(2)       |
| 3. ROM(1)                    | 4. ROM(2)       |
| 5. RAM(1)                    | 6. RAM(2)       |
| 7. I/O PORT(1)               | 8. I/O PORT(2)  |
| 9. INTERRUPT CONTROL(1)      |                 |
| 10. INTERRUPT CONTROL(2)     |                 |
| 11. FDC PORT(1)              | 12. FDC PORT(2) |
| 13. SERIAL INTERFACE(1)      |                 |
| 14. SERIAL INTERFACE(2)      |                 |
| 15. KEYBOARD(1)              | 16. KEYBOARD(2) |
| 17. CRT C(1)                 | 18. CRT C(2)    |
| 19. GVRAM ADDRESS CONTROL(1) |                 |
| 20. GVRAM ADDRESS CONTROL(2) |                 |
| 21. GVRAM 0 AND GVRAM 1(1)   |                 |
| 22. GVRAM 0 AND GVRAM 1(2)   |                 |
| 23. GVRAM 2(1)               | 24. GVRAM 2(2)  |
| 25. COLOR CONTROL(1)         |                 |
| 26. COLOR CONTROL(2)         |                 |
| 27. VIDEO(1)                 | 28. VIDEO(2)    |
| 29. BUS SLOT(1)              | 30. BUS SLOT(2) |
| 31. DMAC                     |                 |



(B)

(7)

(6)

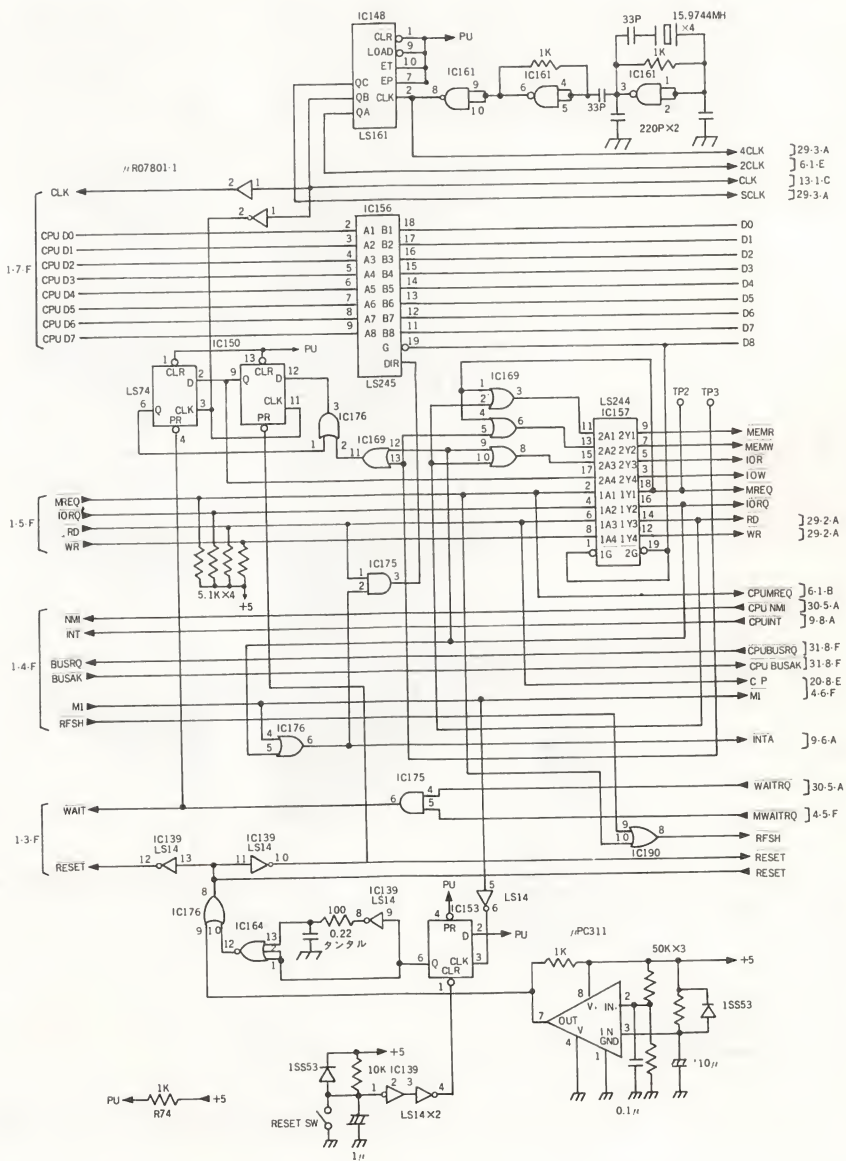
(5)

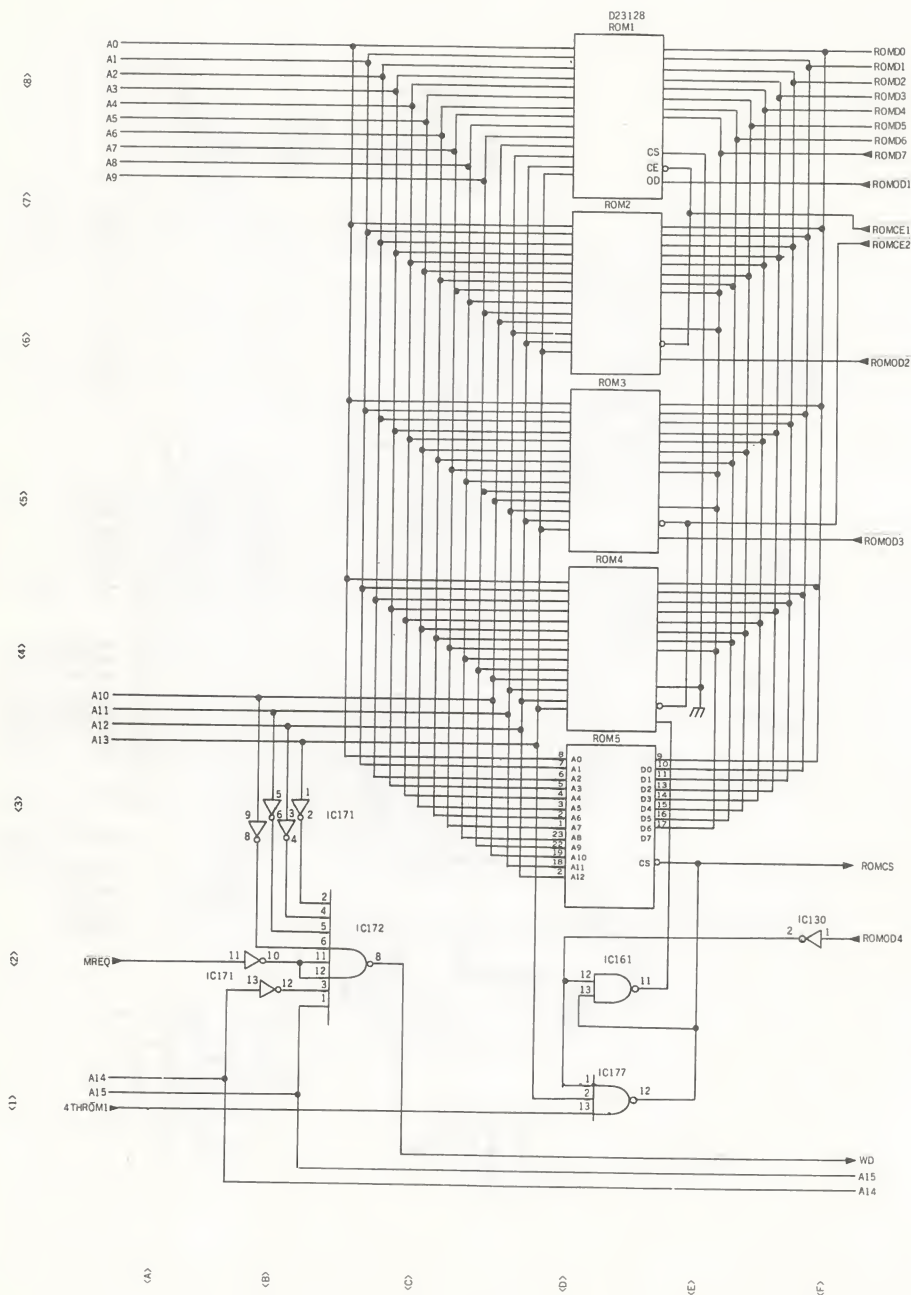
(4)

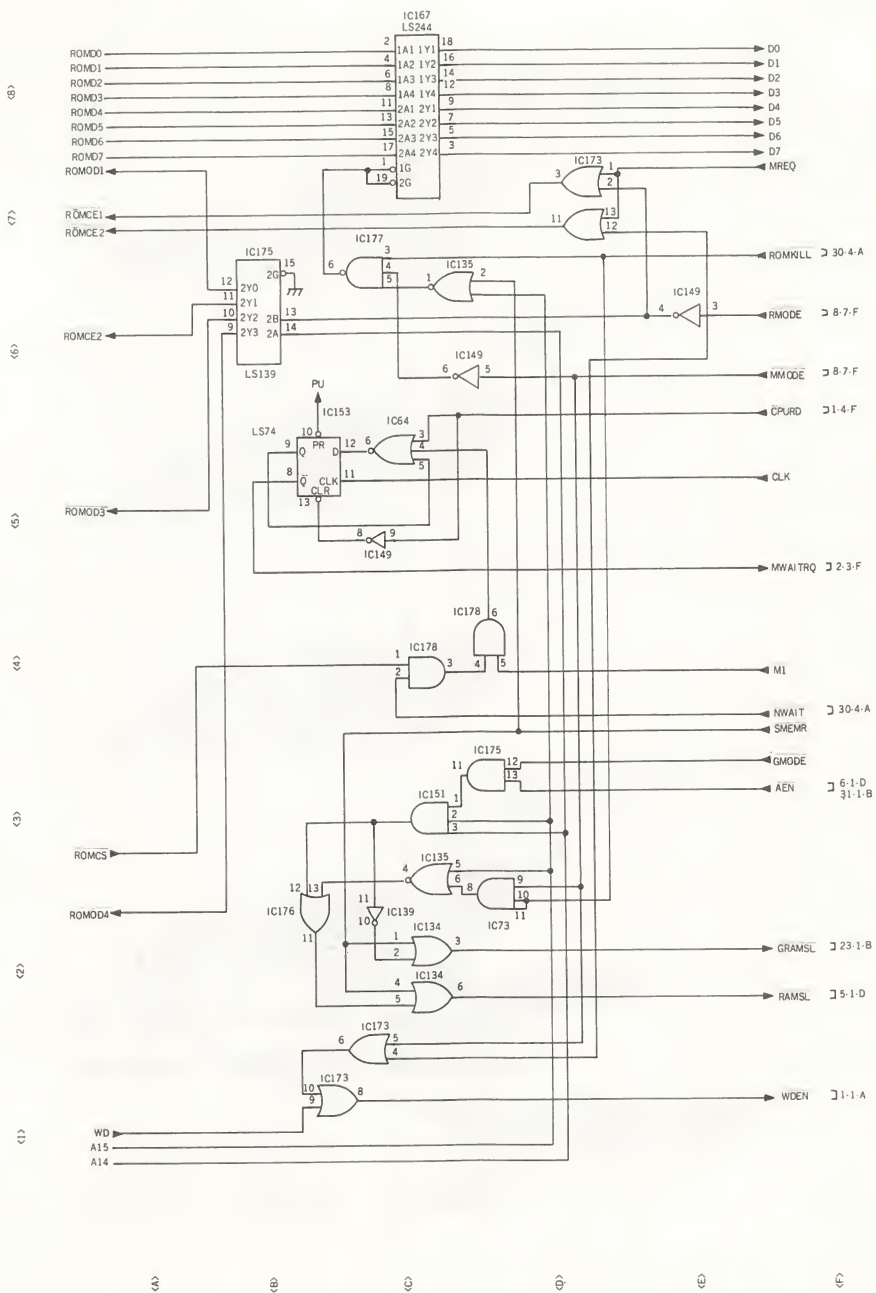
(3)

(2)

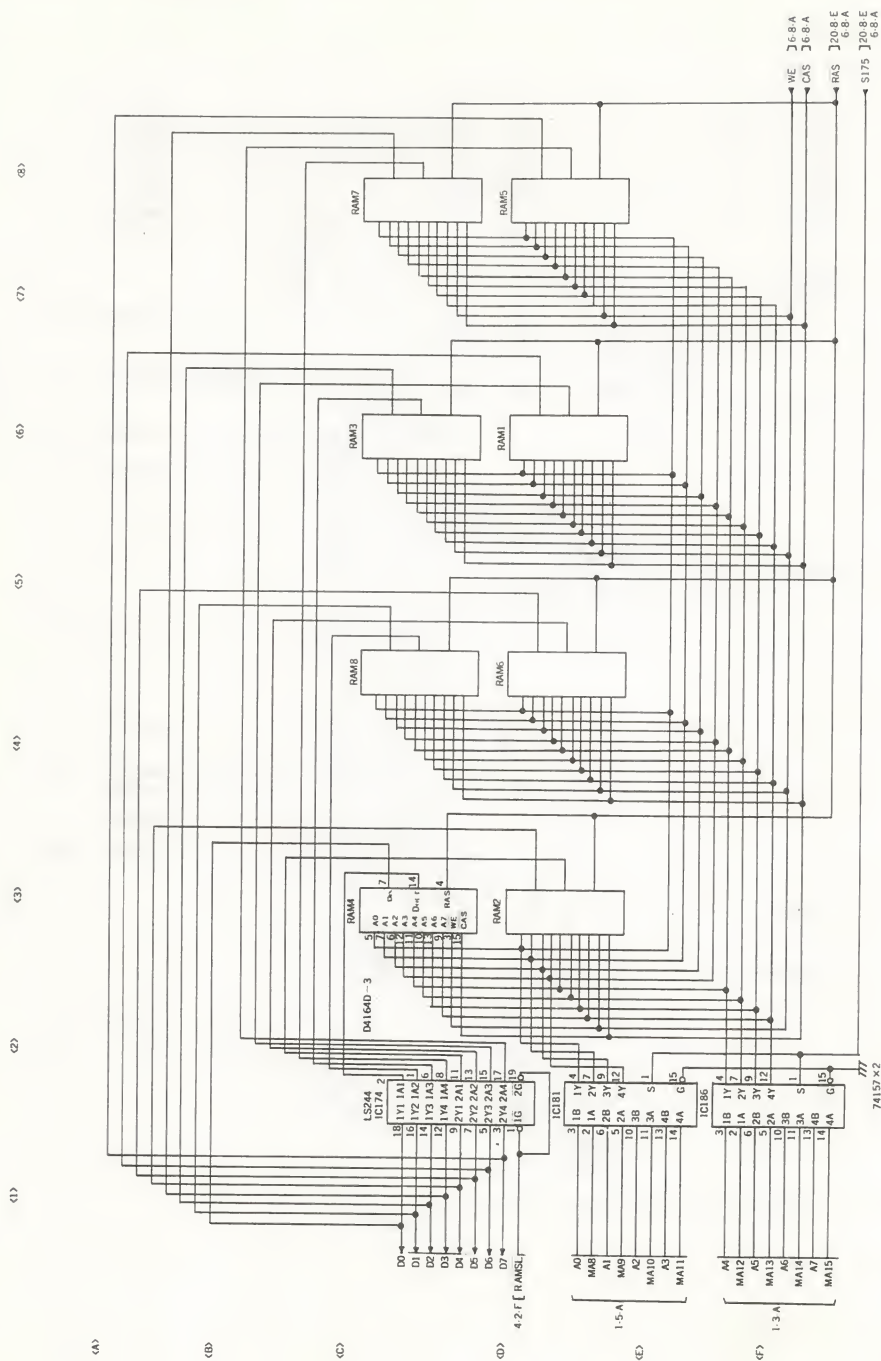
(1)



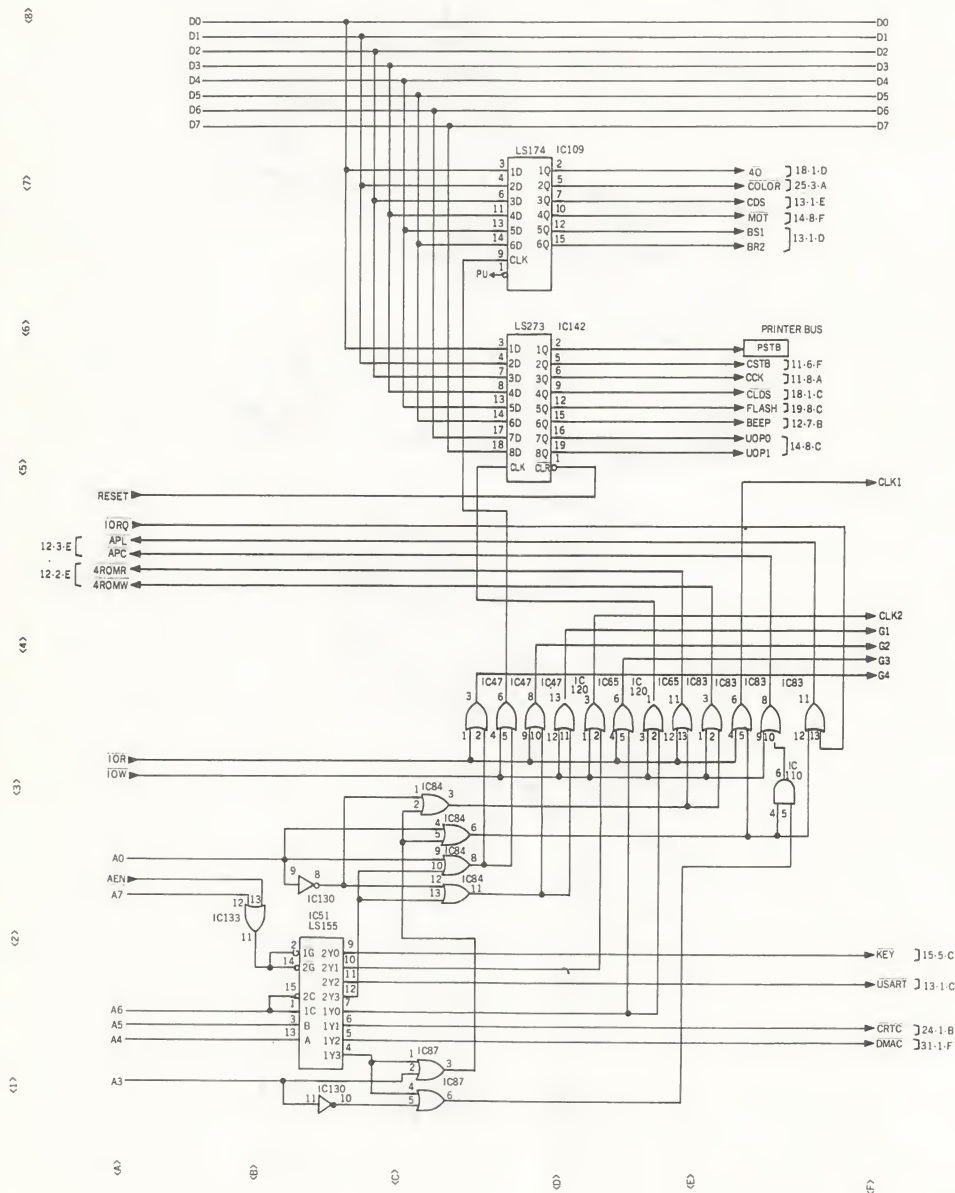


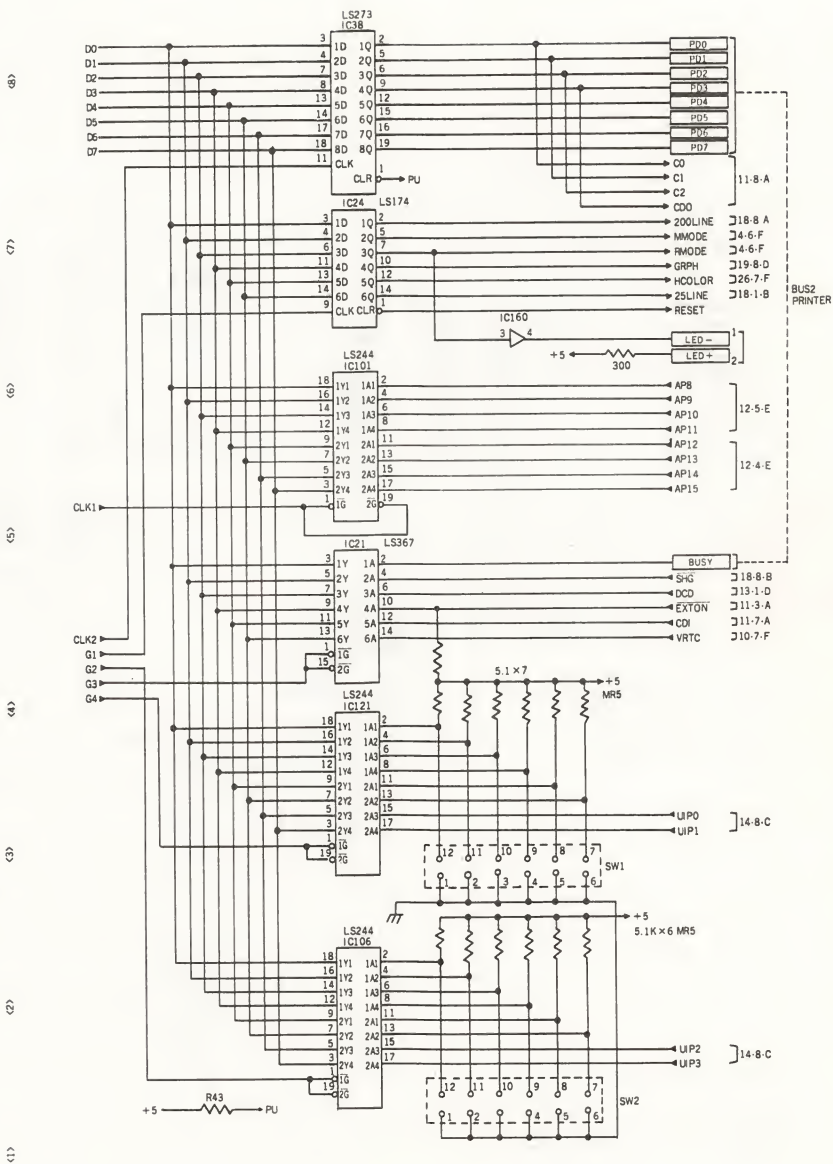


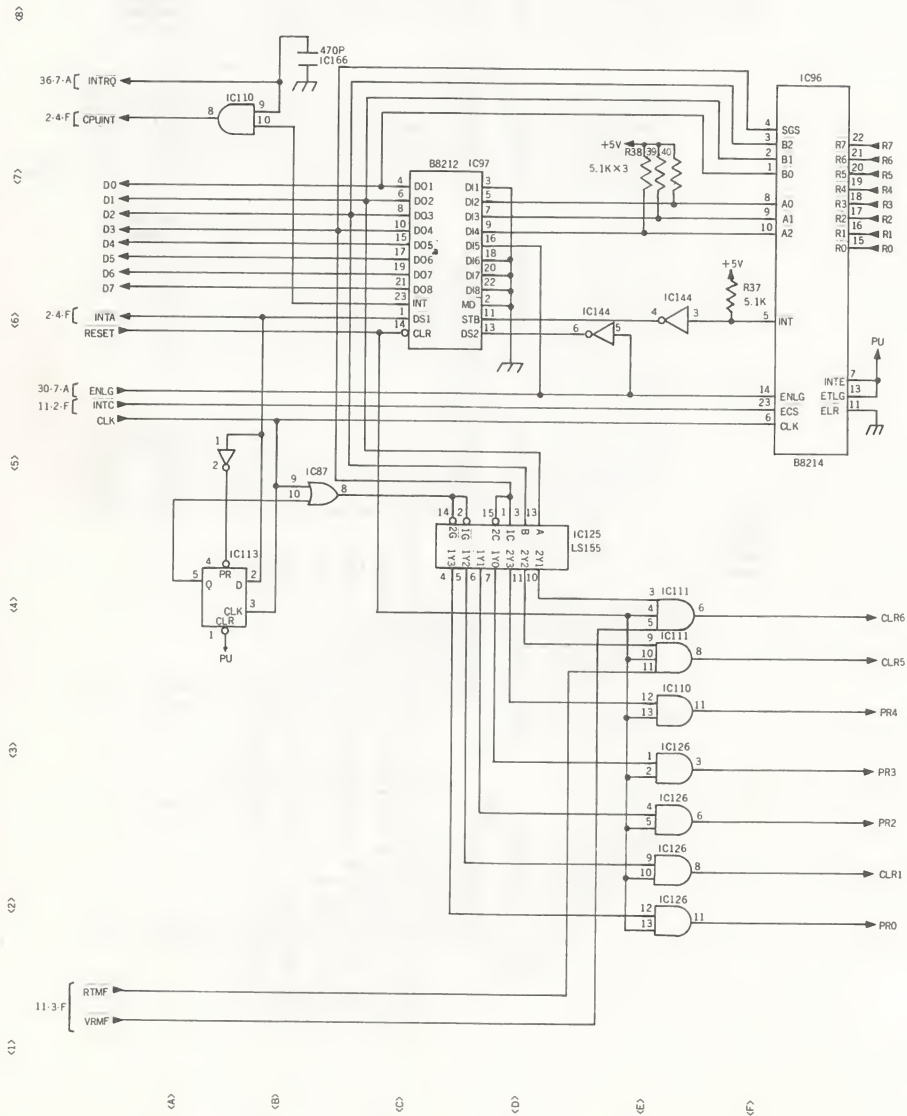












(B)

(7)

(6)

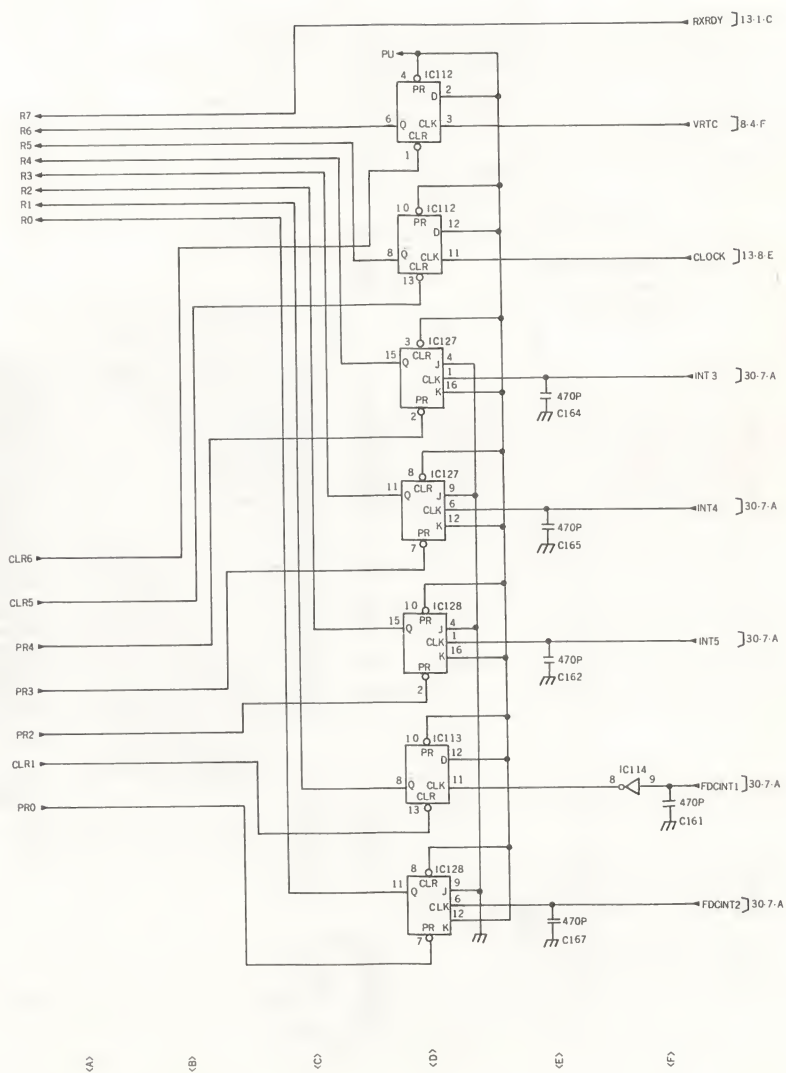
(5)

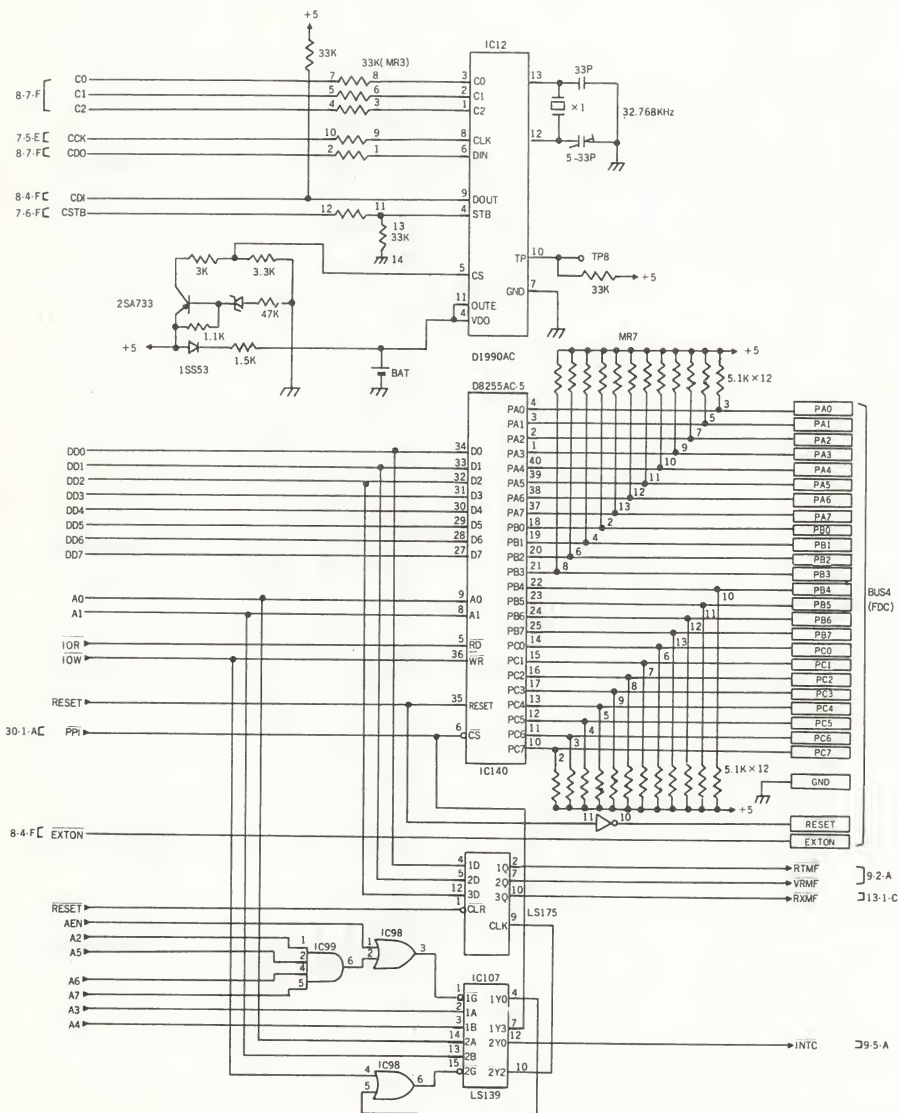
(4)

(3)

(2)

(1)







(8)

(7)

(6)

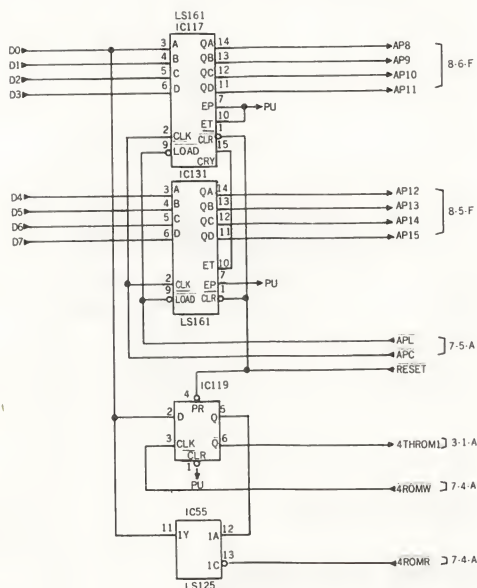
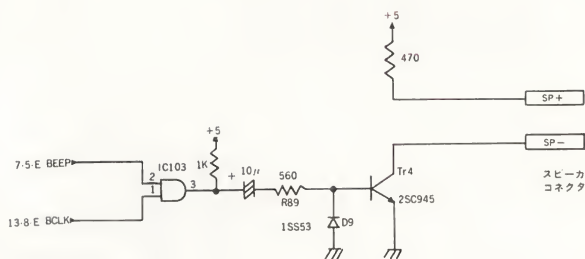
(5)

(4)

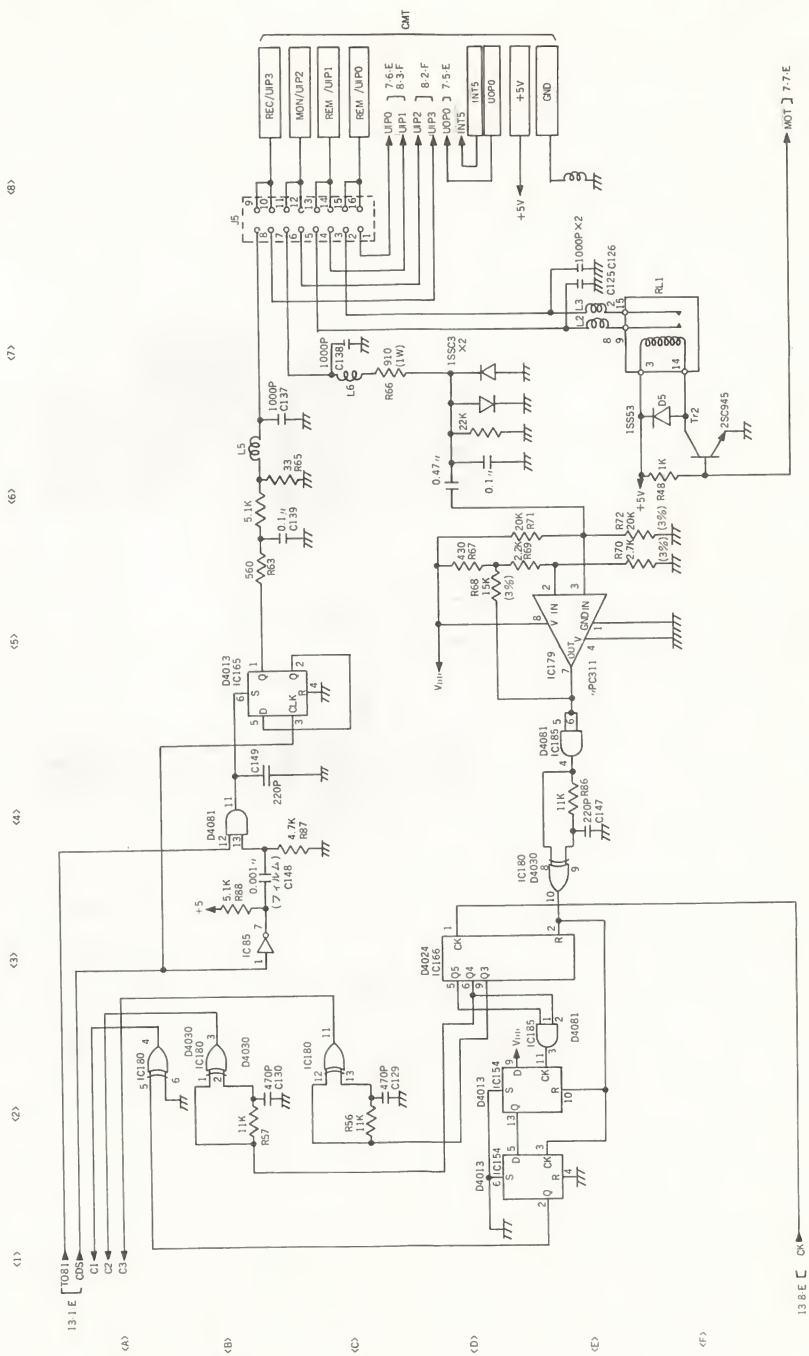
(3)

(2)

(1)







(B)

(7)

(6)

(5)

(4)

(3)

(2)

(1)

(A)

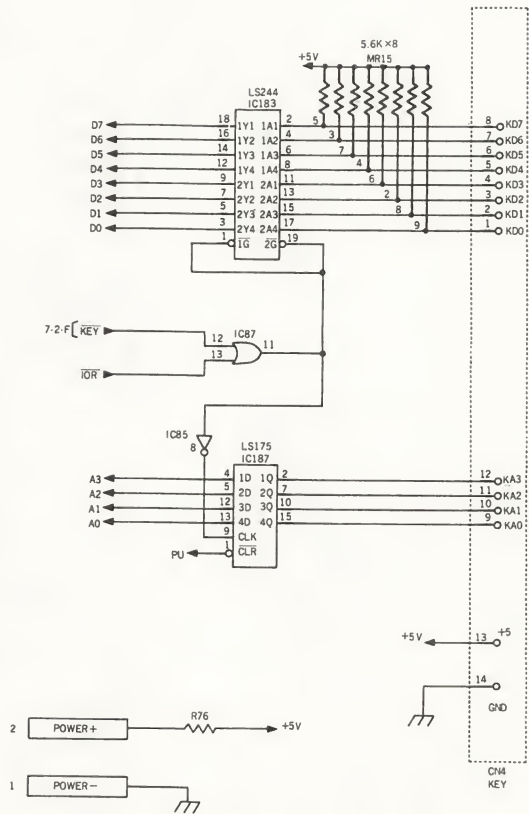
(B)

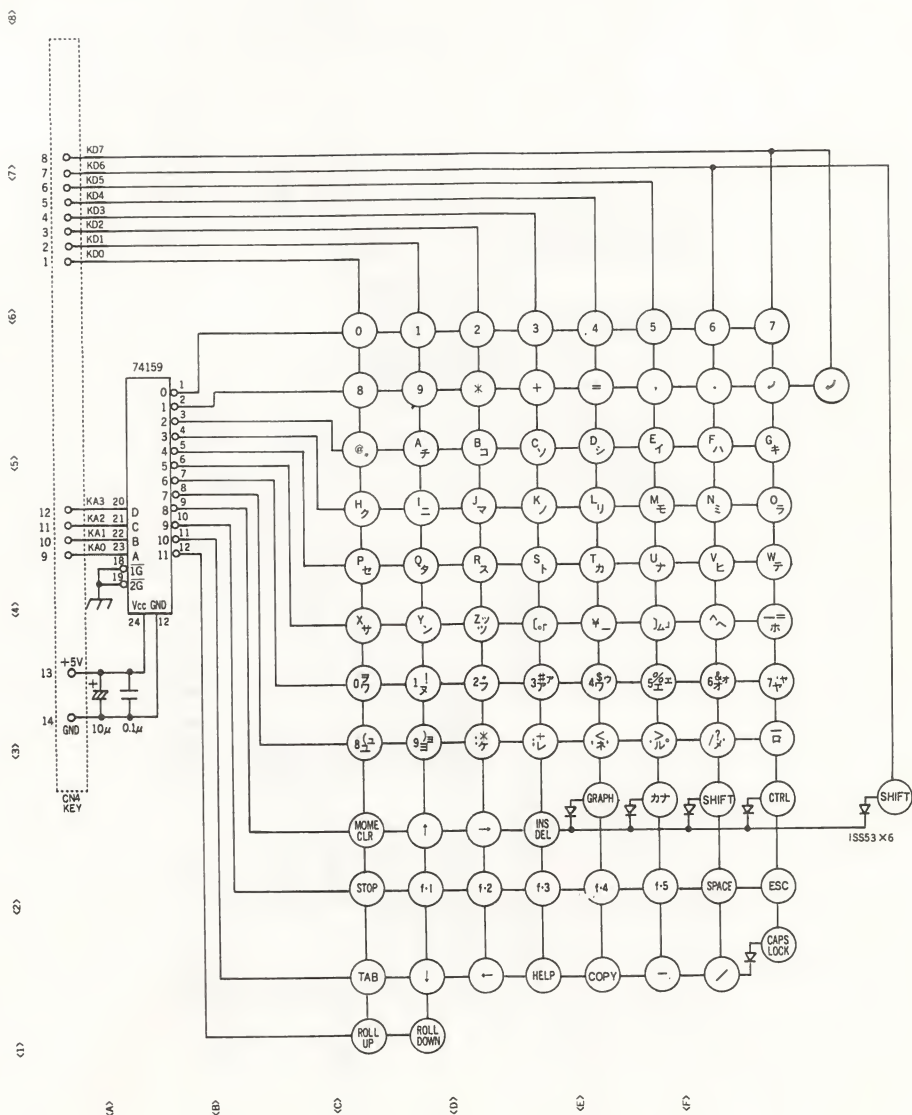
(C)

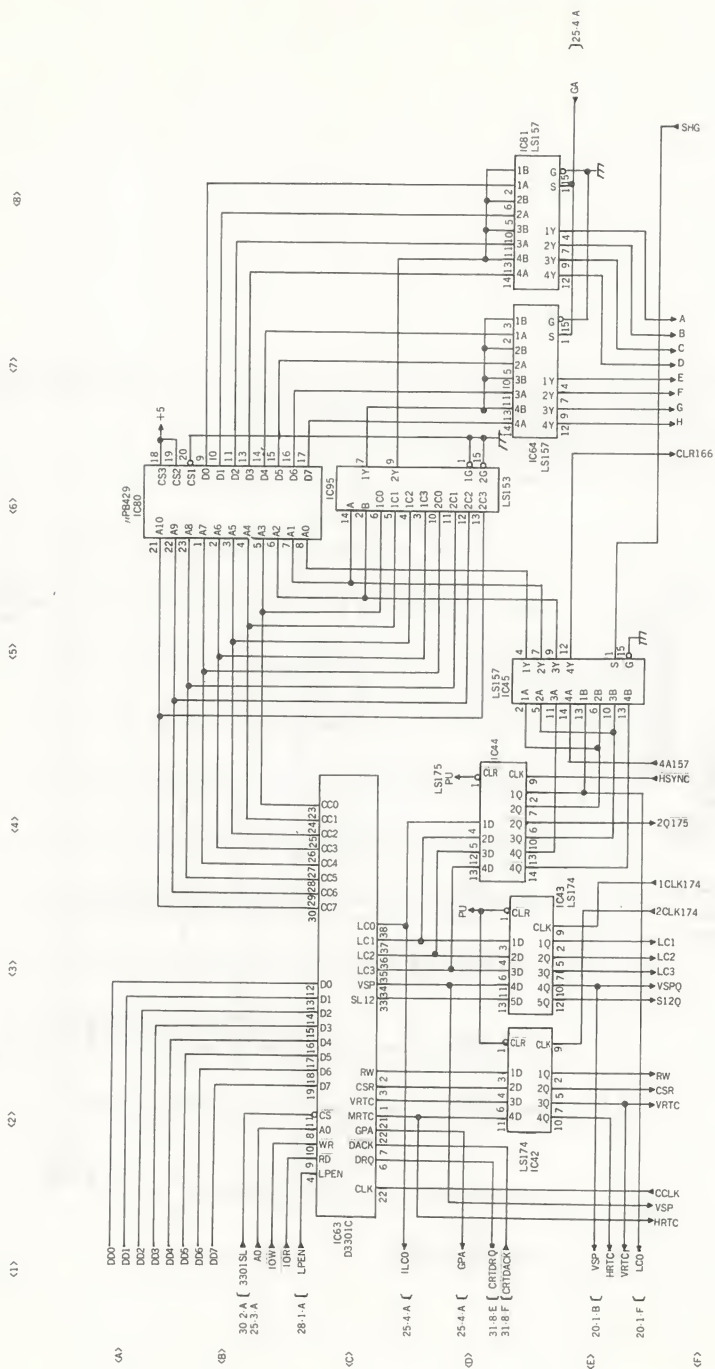
(D)

(E)

(F)

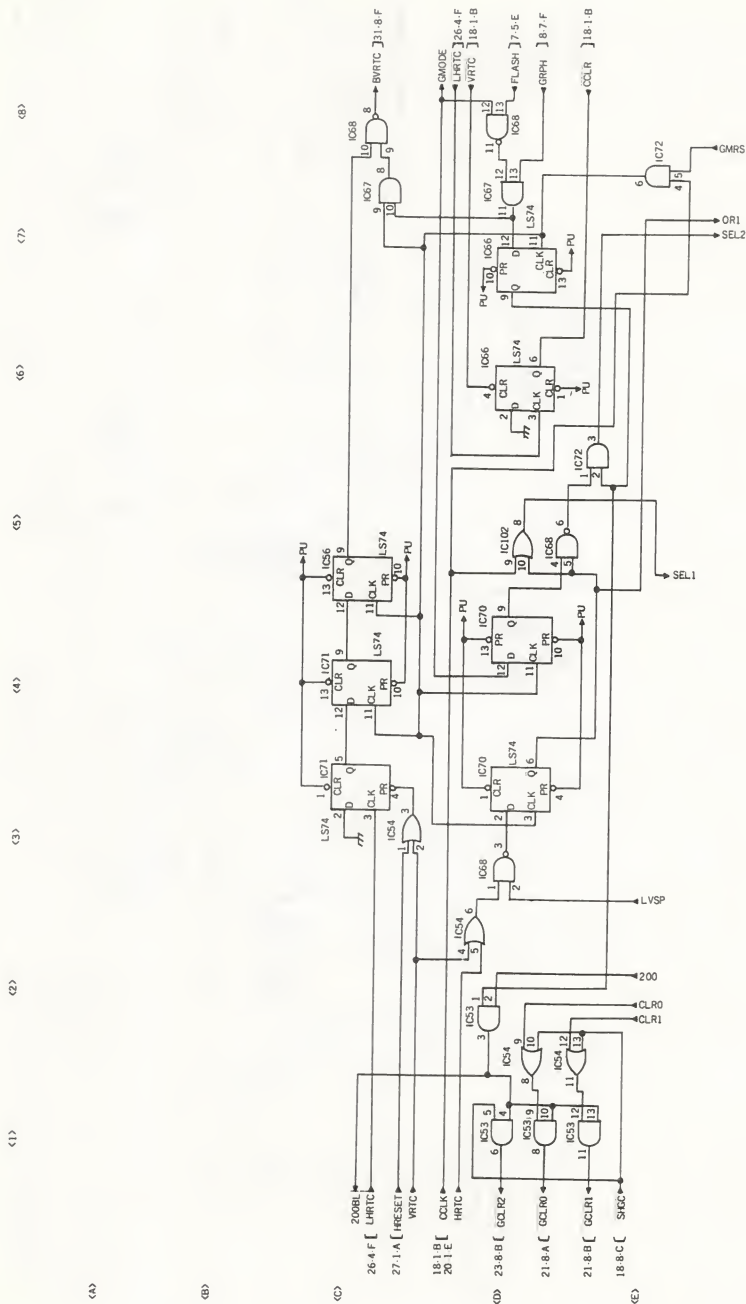






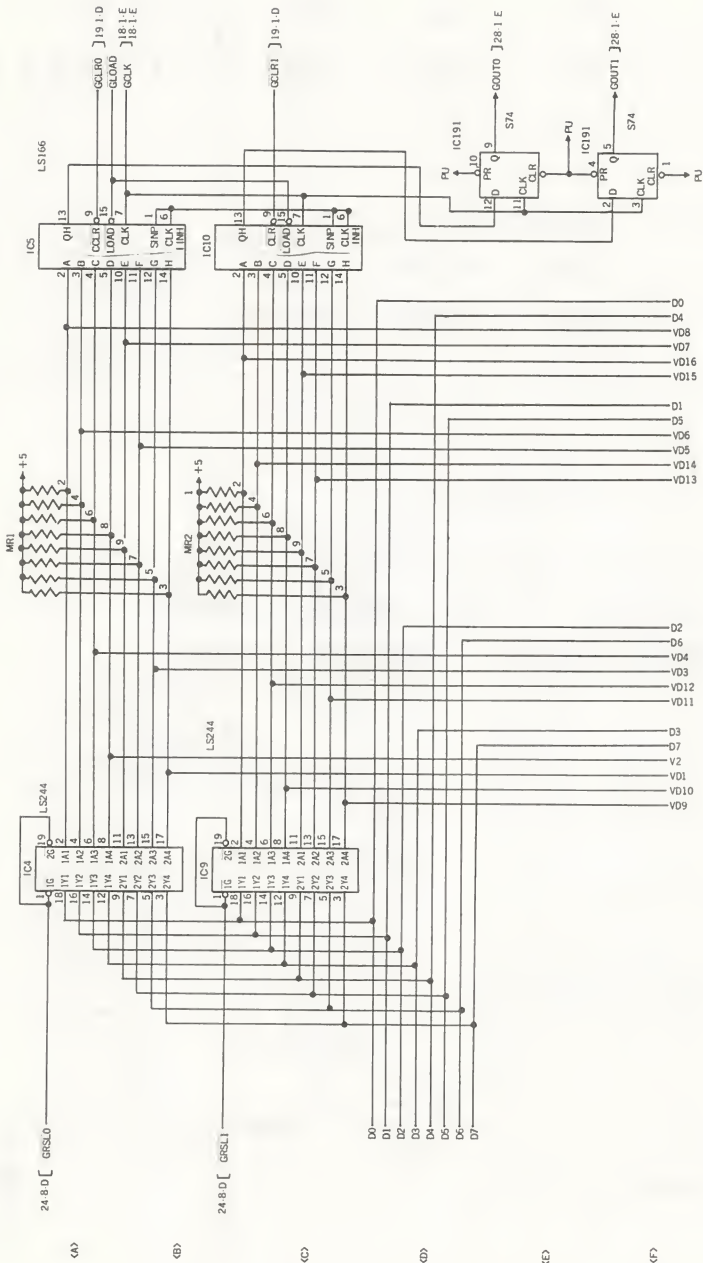


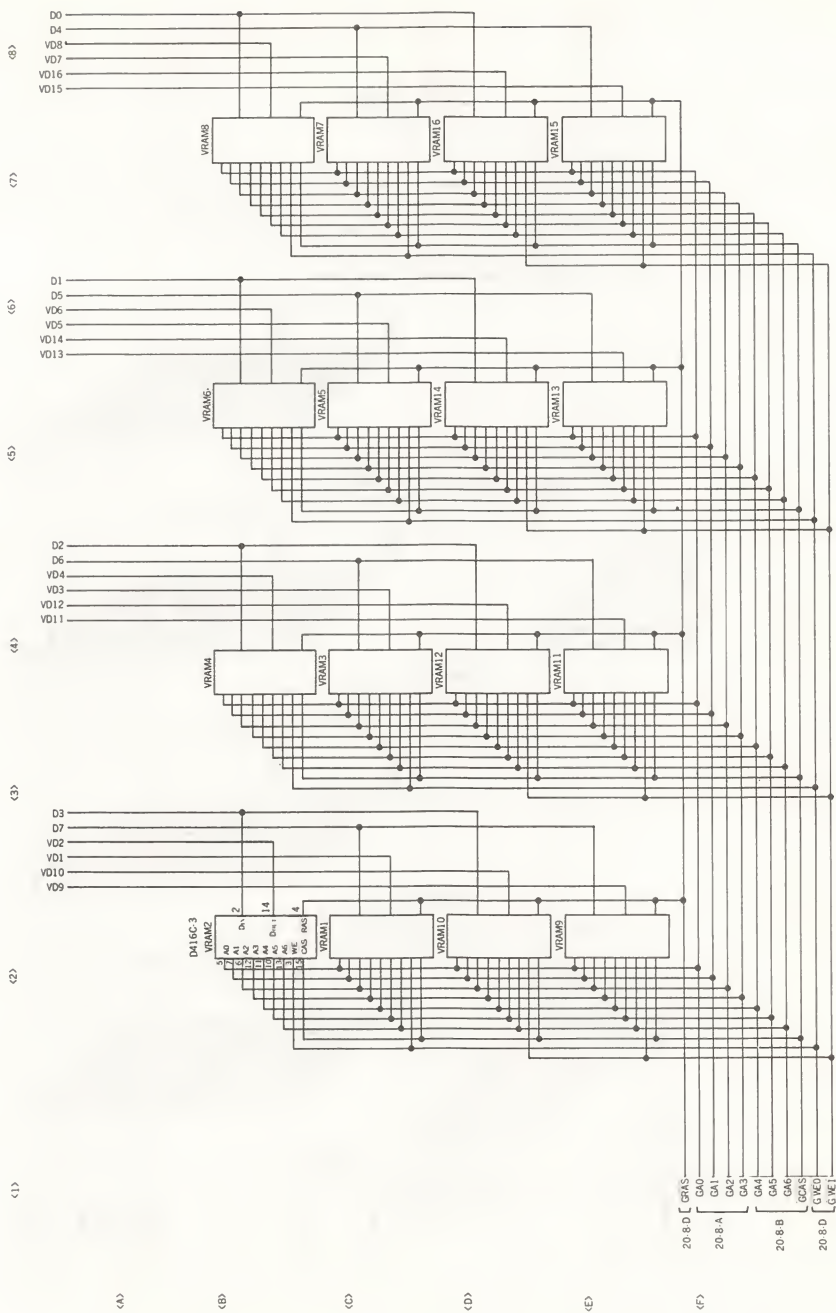


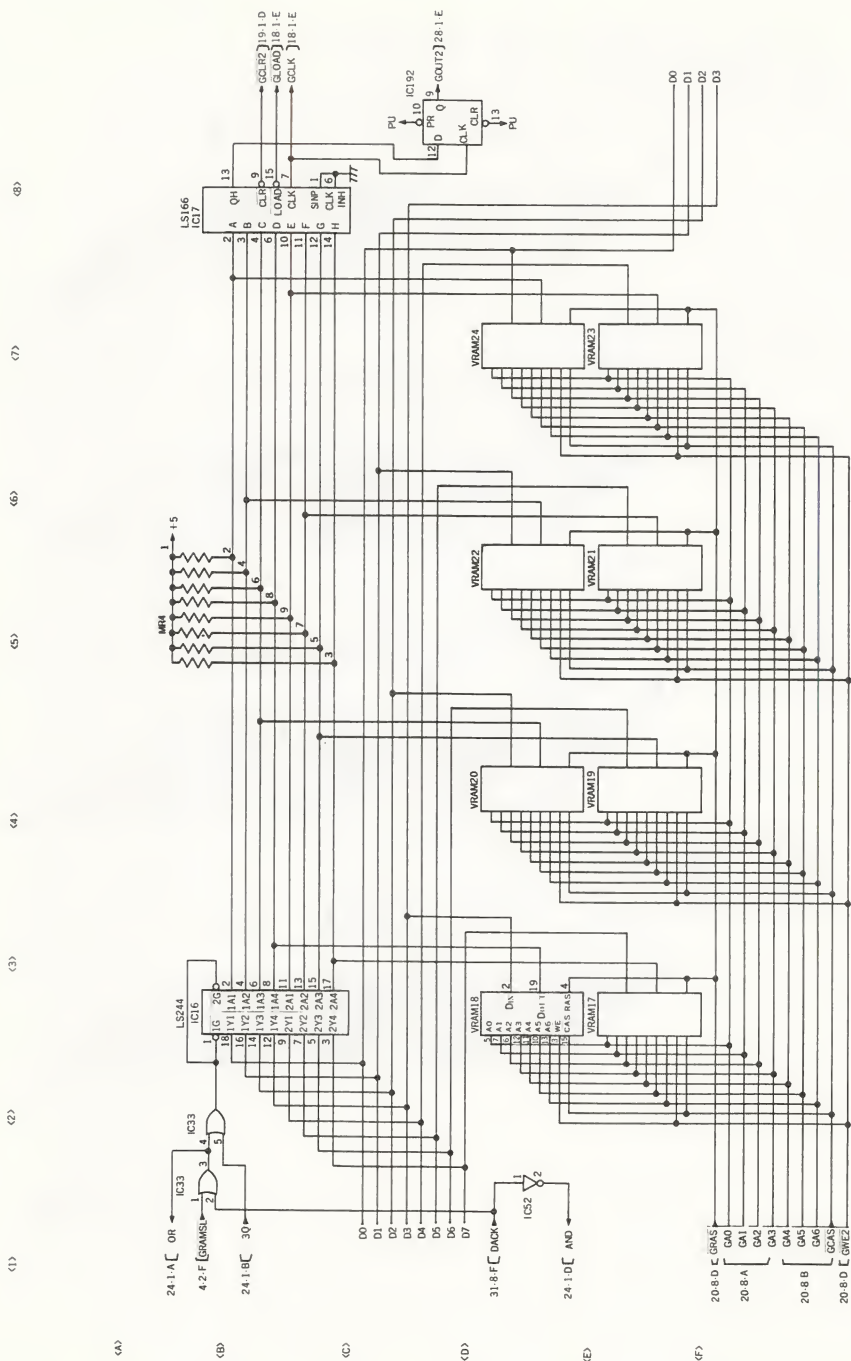


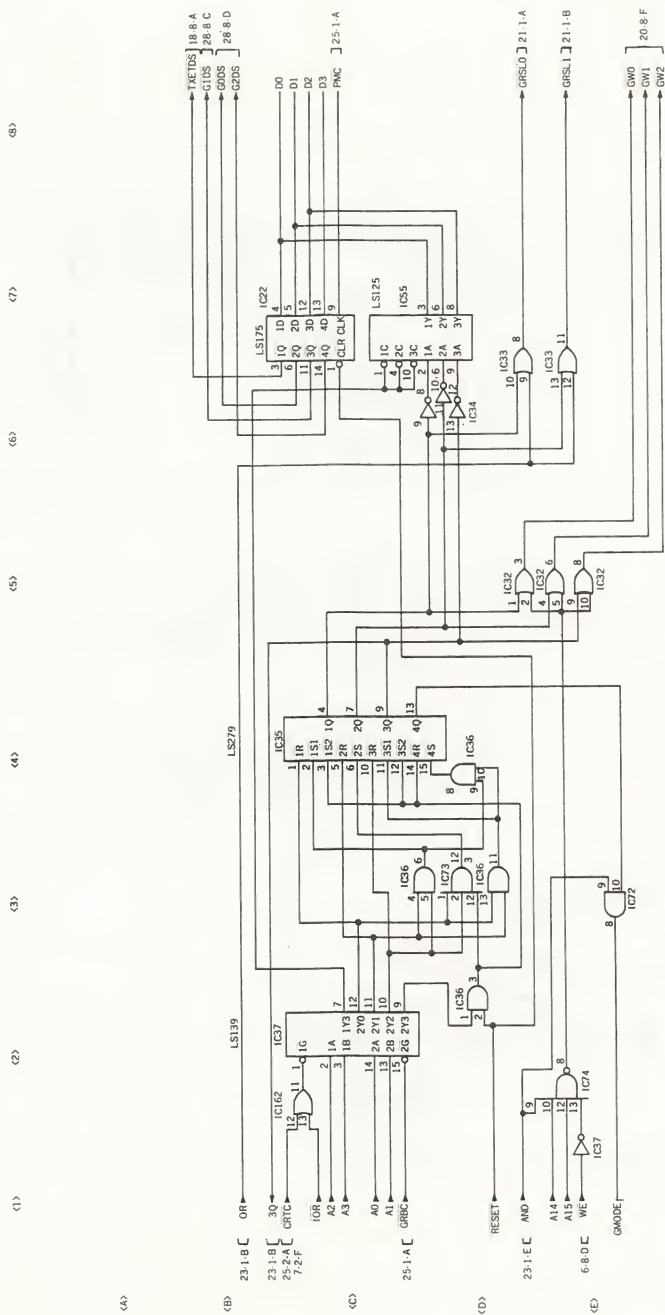


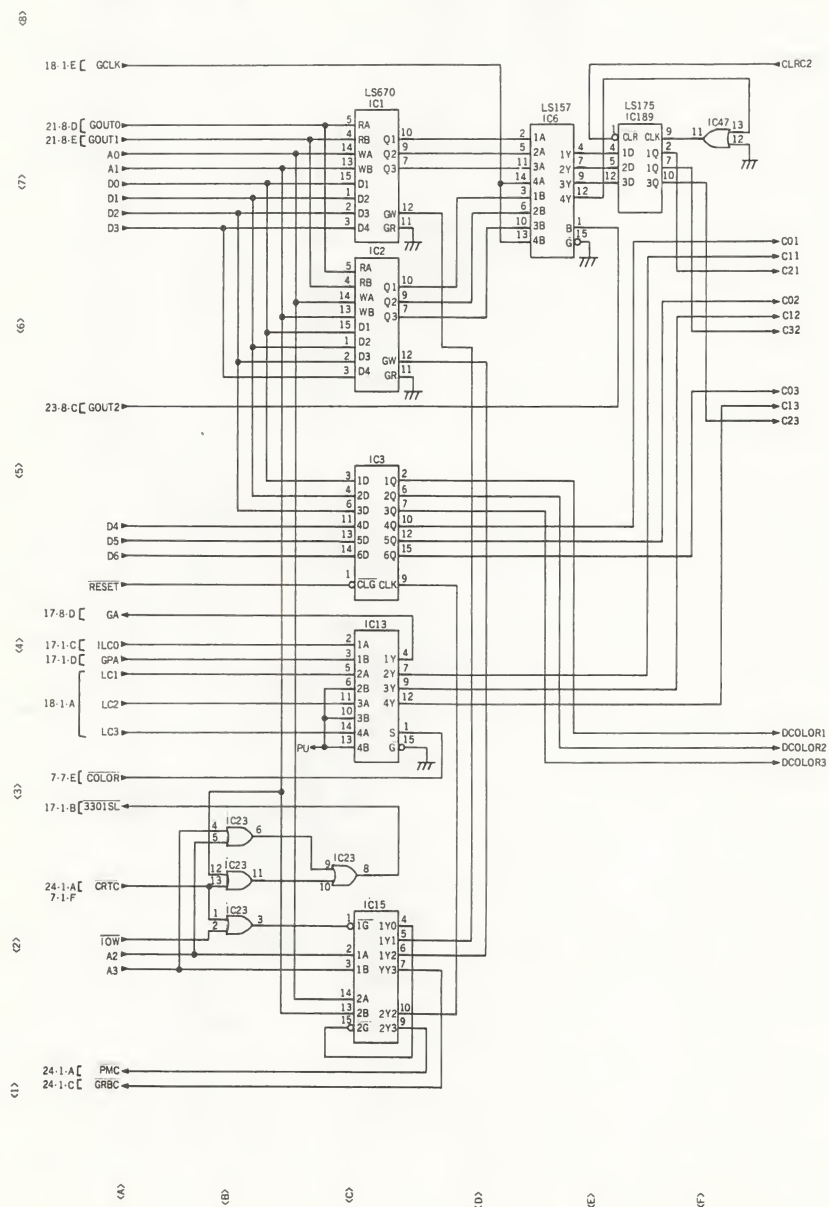
(1) (2) (3) (4) (5) (6) (7) (8)





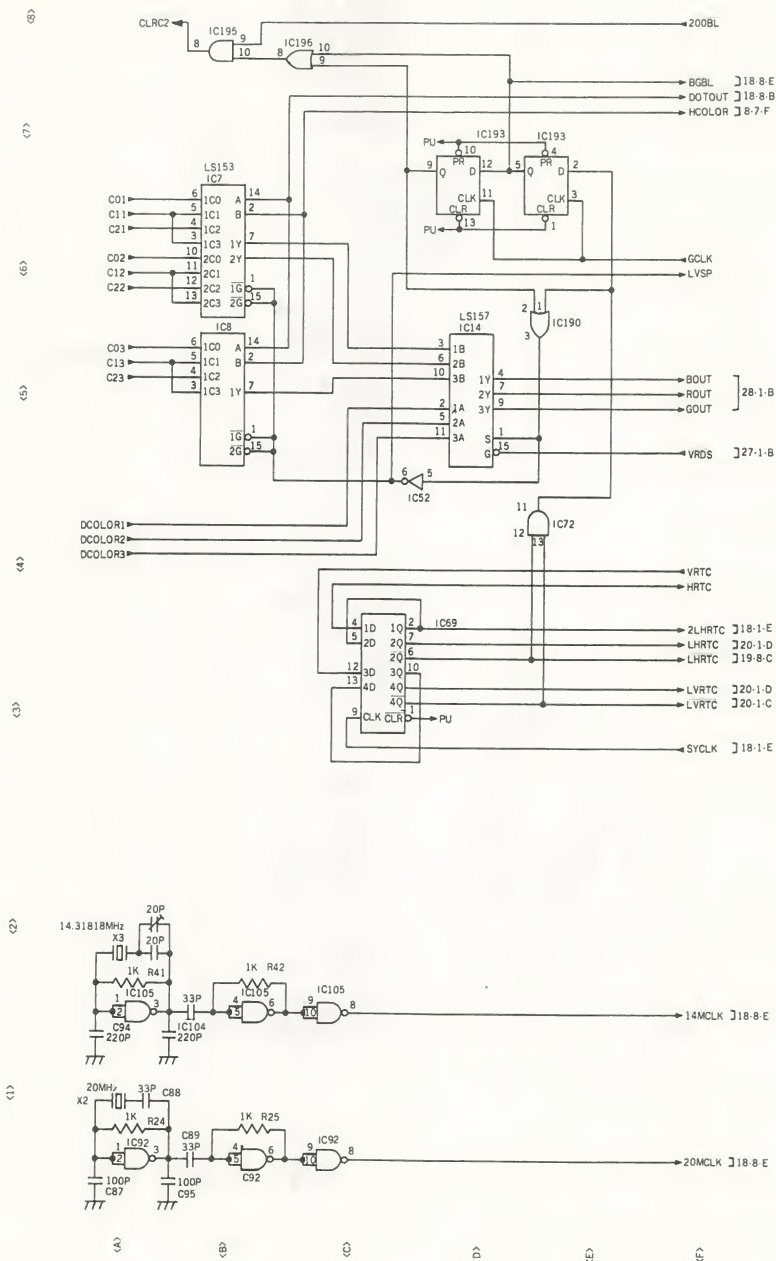




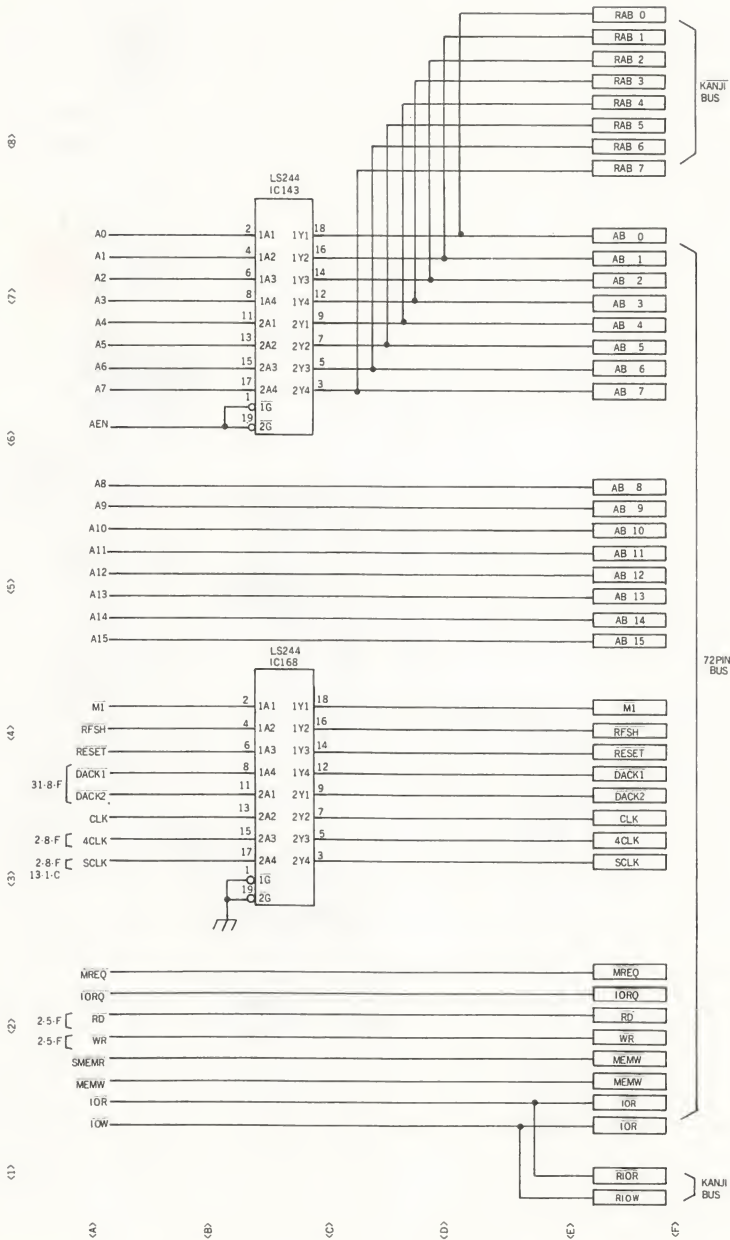


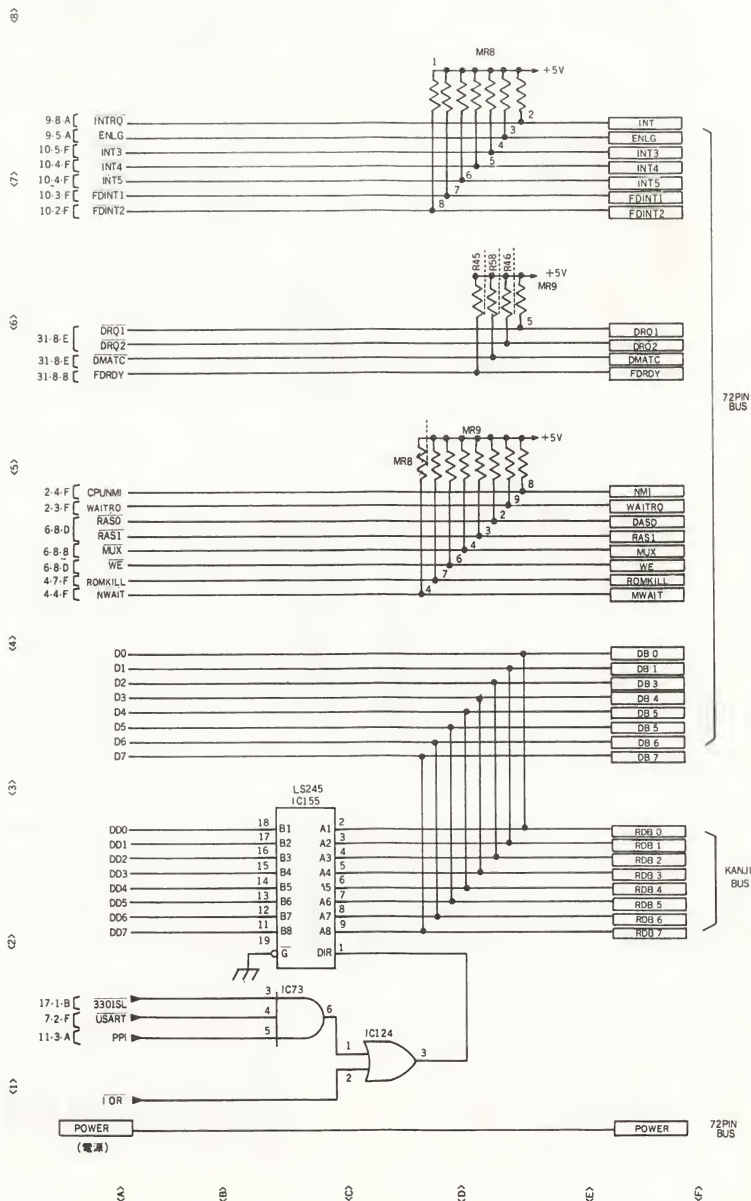














## ***APPENDIX***





## A . 中間言語，処理アドレス一覧表

8 1 H	END	5 0 E 5 H
8 2 H	FOR	0 8 B F H
8 3 H	NEXT	5 2 B D H
8 4 H	DATA	0 C 7 7 H
8 5 H	INPUT	1 0 2 D H
8 6 H	DIM	5 A C 5 H
8 7 H	READ	1 0 F 9 H
8 8 H	LET	0 C 9 C H
8 9 H	GOTO	0 B F 9 H
8 A H	RUN	0 B 7 C H
8 B H	IF	0 E 0 5 H
8 C H	RESTORE	5 0 A 5 H
8 D H	GOSUB	0 B B F H
8 E H	RETURN	0 C 4 1 H
8 F H	REM	0 C 7 9 H
9 0 H	STOP	5 0 C A H
9 1 H	PRINT	0 E 5 4 H
9 2 H	CLEAR	5 2 2 E H
9 3 H	LIST	1 8 D 9 H
9 4 H	NEW	7 7 D D H
9 5 H	ON	0 D 0 1 H
9 6 H	WAIT	1 8 0 0 H
9 7 H	DEF	1 5 D 7 H
9 8 H	POKE	1 B 8 4 H
9 9 H	CONT	5 1 4 0 H
9 A H	OUT	1 7 F A H
9 B H	LPRINT	0 E 4 C H
9 C H	LLIST	1 8 D 4 H
9 D H	CONSOLE	7 0 7 1 H
9 E H	WIDTH	1 8 1 A H
9 F H	ELSE	0 C 7 9 H
A 0 H	TRON	5 1 5 8 H
A 1 H	TROFF	5 1 5 9 H
A 2 H	SWAP	5 1 5 E H
A 3 H	ERASE	5 1 9 C H
A 4 H	EDIT	6 5 7 B H
A 5 H	ERROR	0 D C A H

A 6 H	RESUME	0 D 8 D H
A 7 H	DELETE	1 B 4 0 H
A 8 H	AUTO	0 D D 5 H
A 9 H	RENUM	<u>7 5 D D H</u> (下線はサブROMのアドレス)
AAH	DEFSTR	0 A C 4 H
ABH	DEFINT	0 A C 7 H
ACH	DEFSNG	0 A C A H
ADH	DEFDBL	0 A C D H
AEH	LINE	0 F A A H
AFH	WHILE	A 8 7 5 H
B 0 H	WEND	A 8 9 F H
B 1 H	CALL	A 9 1 6 H
B 5 H	WRITE	A C 7 5 H
B 6 H	COMMON	A C 7 2 H
B 7 H	CHAIN	A 9 8 C H
B 8 H	OPTION	1 C 8 9 H
B 9 H	RANDOMIZE	1 C D 1 H
BAH	DSKO\$	A 1 C 4 H
BBH	OPEN	4 7 9 8 H
BCH	FIELD	4 A 5 C H
BDH	GET	7 1 9 8 H
BEH	PUT	7 1 A 6 H
BFH	SET	9 D A D H
C 0 H	CLOSE	4 B 0 4 H
C 1 H	LOAD	4 8 5 4 H
C 2 H	MERGE	4 8 5 5 H
C 3 H	FILES	9 F 2 F H
C 4 H	NAME	9 9 B 6 H
C 5 H	KILL	9 B A D H
C 6 H	LSET	4 9 A B H
C 7 H	RSET	4 9 A A H
C 8 H	SAVE	4 8 A 3 H
C 9 H	LFILES	9 F 2 A H
CAH	MON	E 8 2 6 H
CBH	COLOR	<u>6 8 7 8 H</u>
CCH	CIRCLE	<u>7 D 9 6 H</u>
CDH	COPY	<u>7 0 5 3 H</u>
CEH	CLS	<u>7 1 B 5 H</u>
CFH	PSET	<u>7 E 0 0 H</u>
D 0 H	PRESET	<u>7 D F B H</u>

D1H	PAINT	<u>7 6 7 4 H</u>
D2H	TERM	<u>7 3 6 7 H</u>
D3H	SCREEN	<u>6 9 8 F H</u>
D4H	BLOAD	A 7 A 8 H
D5H	BSAVE	A 7 3 D H
D6H	LOCATE	7 1 4 E H
D7H	BEEP	3 E B 4 H
D8H	ROLL	8 8 0 3 H
D9H	HELP	7 2 A B H
DBH	KANJ I	
DCH	TO	
DDH	THEN	
DEH	TAB	
DFH	STEP	
E0H	USR	1 5 8 F H
E1H	FN	1 6 0 0 H
E2H	SPC	
E3H	NOT	
E4H	ERL	1 3 A 2 H
E5H	ERR	1 3 9 4 H
E6H	STRING\$	5 7 2 2 H
E7H	USING	6 6 4 2 H
E8H	INSTR	5 7 D 7 H
EAH	VARPTR	1 3 B 0 H
EBH	ATTR\$	9 E 1 D H
ECH	DSKI\$	A 1 8 5 H
EDH	SRQ	
EEH	OFF	
EFH	INKEY\$	5 A A 3 H
F0H	>	
F1H	=	
F2H	<	
F3H	+	
F4H	-	
F5H	*	
F6H	/	
F7H	^	
F8H	AND	
F9H	OR	
FAH	XOR	

FBH		EQV		
FCH		IMP		
FDH		MOD		
FEH		¥		
FFH	81H	LEFT\$	575AH	
FFH	82H	RIGHT\$	578AH	
FFH	83H	MID\$	585AH	5793H (関数)
FFH	84H	SGN	20B3H	
FFH	85H	INT	2295H	
FFH	86H	ABS	20A0H	
FFH	87H	SQR	2E05H	9650H (倍精度)
FFH	88H	RND	2F1AH	2F1AH (倍精度)
FFH	89H	SIN	2F91H	9615H (倍精度)
FFH	8AH	LOG	1F10H	956FH (倍精度)
FFH	8BH	EXP	2E6EH	94F0H (倍精度)
FFH	8CH	COS	2F8BH	94E7H (倍精度)
FFH	8DH	TAN	302CH	9691H (倍精度)
FFH	8EH	ATN	89B3H	948BH (倍精度)
FFH	8FH	FRE	58E4H	
FFH	90H	INP	17E5H	
FFH	91H	POS	1586H	
FFH	92H	LEN	56F8H	
FFH	93H	STR\$	54CBH	
FFH	94H	VAL	57B4H	
FFH	95H	ASC	5704H	
FFH	96H	CHR\$	5714H	
FFH	97H	PEEK	1B7AH	
FFH	98H	SPACE\$	5741H	
FFH	99H	OCT\$	54C1H	
FFH	9AH	HEX\$	54C6H	
FFH	9BH	LPOS	1581H	
FFH	9CH	CINT	21A0H	
FFH	9DH	CSNG	2214H	
FFH	9EH	CDBL	223EH	
FFH	9FH	FIX	2286H	
FFH	A0H	CVI	4ABAH	
FFH	A1H	CVS	4ABDH	
FFH	A2H	CVD	4AC0H	
FFH	A3H	EOF	4C51H	A685H
FFH	A4H	LOC	4C2FH	A640H

FFH	A5H	LOF	4C40H	A669H
FFH	A6H	FPOS	4C62H	A6C9H
FFH	A7H	MKI\$	4AA1H	
FFH	A8H	MKS\$	4AA4H	
FFH	A9H	MKD\$	4AA7H	
FFH	D0H	DSKF	8EC3H	
FFH	D1H	VIEW	<u>6AC6H</u>	<u>6CA8H</u> (関数)
FFH	D2H	WINDOW	<u>6C55H</u>	<u>6DC5H</u> (関数)
FFH	D3H	POINT	<u>6E25H</u>	<u>6DC0H</u> (関数)
FFH	D4H	CSRLIN	3F31H	
FFH	D5H	MAP	<u>6D29H</u>	
FFH	D6H	SEARCH	88E9H	
FFH	D7H	MOTOR	7F30H	
FFH	D8H	PEN	72C8H	<u>72EDH</u> (関数)
FFH	D9H	DATE\$	721CH	<u>74EEH</u> (関数)
FFH	DAH	COM	7D71H	
FFH	DBH	KEY	<u>742EH</u>	
FFH	DCH	TIMES	<u>7279H</u>	<u>752AH</u> (関数)
FFH	DDH	WBYTE	EEA1H	
FFH	DEH	RBYTE	EEA4H	
FFH	DFH	POLL	EEA7H	
FFH	E0H	ISSET	EEAAH	
FFH	E1H	IEEE	EEB9H	
FFH	E2H	IRESET	EEADH	
FFH	E3H	STATUS	EEB0H	EEB3H (関数)
FFH	E4H	CMD	EEB6H	

テーブルは69FEH～6F11H番地です。

## B. 機械語モニタ処理アドレス一覧表

A	(アセンブル)		6 D 0 2 H
B	(ベース)		6 9 5 E H
D	(ダンプ・メモリ)		6 1 A 3 H
E	(エディット・メモリ)		6 4 D 8 H
F	(フィル・メモリ)		6 3 D 8 H
G	(ゴー)		6 4 0 6 H
I	(インプット)		6 4 7 9 H
L	(ディス・アセンブル)		6 9 7 6 H
M	(ムーブ・メモリ)		6 4 A 6 H
O	(アウトプット)		6 4 9 1 H
P	(プリンタ・スイッチ)		6 E 7 3 H
R	(リード・テープ)		6 8 0 7 H
S	(セット・メモリ)		6 1 3 5 H
TM	(テスト・メモリ)		7 2 7 C H
V	(ベリファイ・テープ)		6 8 0 6 H
W	(ライト・テープ)		6 7 5 4 H
X	(イグザミン・レジスタ)		6 2 5 4 H
CTRL-A	(ヘルプ)		7 4 6 F H
CTRL-B	(リターン)		6 E 5 F H
CTRL-D	(ダンプ・ディスク)	E 6 7 2 H	8 4 2 7 H
CTRL-R	(リード・ディスク)	E 6 7 8 H	8 4 F 8 H
CTRL-W	(ライト・ディスク)	E 6 7 5 H	8 4 F 1 H

テーブルは 6 0 F 0 H ~ 6 1 0 5 H 番地です。



# C. $\mu$ PD 3301 アトリビュート・コード表

モード ビット		カ ラ ー モ ー ド		白黒モード
		カラー指定	機能指定	
0	0		ノ ー マ ル	ノ ー マ ル
	1		シークレット	シークレット
1	0		ノ ー マ ル	ノ ー マ ル
	1		ブ リ ン ク	ブ リ ン ク
2	0		ノ ー マ ル	ノ ー マ ル
	1		リ バ ー ス	リ バ ー ス
3	0	機 能 指 定		
	1	カ ラ ー 指 定		
4	0	キ ャ ラ ク タ	ノ ー マ ル	ノ ー マ ル
	1	グ ラ フ ィ ッ ク	オーバー・ライン	オーバー・ライン
5	0	B信号 OFF	ノ ー マ ル	ノ ー マ ル
	1	B信号 O N	アンダー・ライン	アンダー・ライン
6	0	R信号 OFF		
	1	R信号 O N		
7	0	G信号 OFF		キ ャ ラ ク タ
	1	G信号 O N		グ ラ フ ィ ッ ク

## RGB信号 → カラー変換表

カラー 信号	黒 (ブラック)	青 (ブルー)	赤 (レッド)	紫 (マゼンタ)	緑 (グリーン)	水 (シアン)	黄 (イエロー)	白 (ホワイト)
G	OFF	OFF	OFF	OFF	OFF	O N	O N	O N
R	OFF	OFF	O N	O N	OFF	OFF	O N	O N
B	OFF	O N	OFF	O N	OFF	O N	OFF	O N

# D. キャラクタ・コード表

上位 4 ビット →

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
下位 4 ビット ↓	0		D <sub>E</sub>		0	@	P	、	p		┐		一	タ	ミ	≡	×
	1	S <sub>H</sub>	D <sub>1</sub>	!	1	A	Q	a	q		┐	。	ア	チ	ム	≡	円
	2	S <sub>X</sub>	D <sub>2</sub>	"	2	B	R	b	r		┐	「	イ	ツ	メ	≡	年
	3	E <sub>X</sub>	D <sub>3</sub>	#	3	C	S	c	s		┐	」	ウ	テ	モ	≡	月
	4	E <sub>T</sub>	D <sub>4</sub>	\$	4	D	T	d	t		┐	、	エ	ト	ヤ	◀	日
	5	E <sub>Q</sub>	N <sub>K</sub>	%	5	E	U	e	u		┐	・	オ	ナ	ユ	◀	時
	6	A <sub>K</sub>	S <sub>N</sub>	&	6	F	V	f	v		┐	ヲ	カ	ニ	ヨ	◀	分
	7	B <sub>L</sub>	E <sub>B</sub>	/	7	G	W	g	w		┐	ア	キ	ヌ	ラ	◀	秒
	8	B <sub>S</sub>	C <sub>N</sub>	(	8	H	X	h	x		┐	イ	ク	ネ	リ	♠	
	9	H <sub>T</sub>	E <sub>M</sub>	)	9	I	Y	i	y		┐	ウ	ケ	ノ	ル	♥	
	A	L <sub>F</sub>	S <sub>B</sub>	*	:	J	Z	j	z		┐	エ	コ	ハ	レ	♦	
	B	H <sub>M</sub>	E <sub>C</sub>	+	;	K	[	k	}		┐	オ	サ	ヒ	ロ	♣	
	C	C <sub>L</sub>	→	,	<	L	¥	l			┐	ヤ	シ	フ	ワ	●	
	D	C <sub>R</sub>	←	—	=	M	]	m	}		┐	ユ	ス	ヘ	ン	○	
	E	S <sub>O</sub>	↑	.	>	N	^	n	~		┐	ヨ	セ	ホ	ゝ	◀	
	F	S <sub>I</sub>	↓	/	?	O	_	o		┐	┐	ツ	ソ	マ	°	◀	

**μCOM-82**

**インストラクション活用表**



本表は、 $\mu\text{COM}-82$ インストラクション活用表(日本電気株式会社)より、転載いたしました。

## 1. $\mu\text{COM}-82$ インストラクション・セット

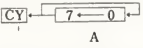
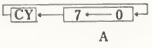
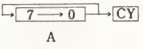
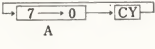
命令群	ニーモニック	オペレーション	フ ラ グ					バイト	ステート	OPコード			
			S	Z	H	P/VN	C			76	543	210	
8 ビット・ロード命令	LD r, r	$r \leftarrow r'$	.	.	.	.	.	1	4	01	r	r	Ⓔ
	LD r, n	$r \leftarrow n$	.	.	.	.	.	2	7	00	r	110	Ⓔ
	LD r, (HL)	$r \leftarrow (\text{HL})$	.	.	.	.	.	1	7	01	r	110	Ⓔ
	LD r, (IX+d)	$r \leftarrow (\text{IX}+d)$	.	.	.	.	.	3	19	11	011	101	Ⓔ
			.	.	.	.	.			01	r	110	Ⓔ
	LD r, (IY+d)	$r \leftarrow (\text{IY}+d)$	.	.	.	.	.	3	19	11	111	101	Ⓔ
			.	.	.	.	.			01	r	110	Ⓔ
	LD (HL), r	$(\text{HL}) \leftarrow r$	.	.	.	.	.	1	7	01	110	r	Ⓔ
	LD (IX+d), r	$(\text{IX}+d) \leftarrow r$	.	.	.	.	.	3	19	11	011	101	Ⓔ
			.	.	.	.	.			01	110	r	Ⓔ
	LD (IY+d), r	$(\text{IY}+d) \leftarrow r$	.	.	.	.	.	3	19	11	111	101	Ⓔ
			.	.	.	.	.			01	110	r	Ⓔ
	LD (HL), n	$(\text{HL}) \leftarrow n$	.	.	.	.	.	2	10	00	110	110	→
	LD (IX+d), n	$(\text{IX}+d) \leftarrow n$	.	.	.	.	.	4	19	11	011	101	→
			.	.	.	.	.			00	110	110	→
	LD (IY+d), n	$(\text{IY}+d) \leftarrow n$	.	.	.	.	.	4	19	11	111	101	→
			.	.	.	.	.			00	110	110	→
	LD A, (BC)	$A \leftarrow (\text{BC})$	.	.	.	.	.	1	7	00	001	010	→
	LD A, (DE)	$A \leftarrow (\text{DE})$	.	.	.	.	.	1	7	00	011	010	→
	LD A, (nn)	$A \leftarrow (\text{nn})$	.	.	.	.	.	3	13	00	111	010	→
16 ビット・ロード命令	LD (BC), A	$(\text{BC}) \leftarrow A$	.	.	.	.	.	1	7	00	000	010	→
	LD (DE), A	$(\text{DE}) \leftarrow A$	.	.	.	.	.	1	7	00	010	010	→
	LD (nn), A	$(\text{nn}) \leftarrow A$	.	.	.	.	.	3	13	00	110	010	→
			.	.	.	.	.			11	101	101	→
	LD A, 1	$A \leftarrow 1$	↑	↑	0	1FF	0	2	9	01	010	111	→
	LD A, R	$A \leftarrow R$	↑	↑	0	1FF	0	2	9	11	101	101	→
			.	.	.	.	.			01	011	111	→
	LD I, A	$I \leftarrow A$	.	.	.	.	.	2	9	11	101	101	→
			.	.	.	.	.			01	000	111	→
	LD R, A	$R \leftarrow A$	.	.	.	.	.	2	9	11	101	101	→
			.	.	.	.	.			01	001	111	→
			.	.	.	.	.			11	101	101	→
16 ビット・ロード命令	LD dd, nn	$dd \leftarrow nn$	.	.	.	.	.	3	10	00	dd0	001	Ⓐ
			.	.	.	.	.			11	101	101	→
	LD IX, nn	$\text{IX} \leftarrow nn$	.	.	.	.	.	4	14	00	100	001	→
			.	.	.	.	.			11	011	101	→
	LD IY, nn	$\text{IY} \leftarrow nn$	.	.	.	.	.	4	14	00	100	001	→
			.	.	.	.	.			11	111	101	→
16 ビット・ロード命令	LD HL, (nn)	$\text{H} \leftarrow (\text{nn}+1)$ $\text{L} \leftarrow (\text{nn})$	.	.	.	.	.	3	16	00	101	010	→
			.	.	.	.	.			11	101	101	→
	LD dd, (nn)	$dd_H \leftarrow (\text{nn}+1)$ $dd_L \leftarrow (\text{nn})$	.	.	.	.	.	4	20	01	dd1	011	Ⓐ
			.	.	.	.	.			11	101	101	→
			.	.	.	.	.			11	101	101	→
			.	.	.	.	.			11	101	101	→

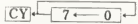
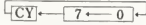
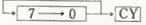
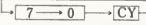
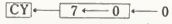
命令群	ニーモニック	オペレーション	フ ラ グ					バイト	ステート	OPコード				
			S	Z	H	P/V	N			C	76	543	210	
16 ビット ・ ロ ー ド 命 令	LD IX, (nn)	$IX_H \leftarrow (nn+1)$ $IX_L \leftarrow (nn)$	・	・	・	・	・	・	4	20	11 011 101 00 101 010 ← n → ← n →			
	LD IY, (nn)	$IY_H \leftarrow (nn+1)$ $IY_L \leftarrow (nn)$	・	・	・	・	・	・	4	20	11 111 101 00 101 010 ← n → ← n →			
	LD (nn), HL	$(nn+1) \leftarrow H$ $(nn) \leftarrow L$	・	・	・	・	・	・	3	16	00 100 010 ← n → ← n →			
	LD (nn), dd	$(nn+1) \leftarrow dd_H$ $(nn) \leftarrow dd_L$	・	・	・	・	・	・	4	20	11 101 101 01 ddo 011 ← n → ← n →	Ⓐ		
	LD (nn), IX	$(nn+1) \leftarrow IX_H$ $(nn) \leftarrow IX_L$	・	・	・	・	・	・	4	20	11 011 101 00 100 010 ← n → ← n →			
	LD (nn), IY	$(nn+1) \leftarrow IY_H$ $(nn) \leftarrow IY_L$	・	・	・	・	・	・	4	20	11 111 101 00 100 010 ← n → ← n →			
	LD SP, HL	$SP \leftarrow HL$	・	・	・	・	・	・	1	6	11 111 001			
	LD SP, IX	$SP \leftarrow IX$	・	・	・	・	・	・	2	10	11 011 101 11 111 001			
	LD SP, IY	$SP \leftarrow IY$	・	・	・	・	・	・	2	10	11 111 101 11 111 001			
	PUSH qq	$(SP-2) \leftarrow qq_L$ $(SP-1) \leftarrow qq_H$	・	・	・	・	・	・	・	1	11	11 qq0 101	Ⓑ	
	PUSH IX	$(SP-2) \leftarrow IX_L$ $(SP-1) \leftarrow IX_H$	・	・	・	・	・	・	・	2	15	11 011 101 11 100 101		
	PUSH IY	$(SP-2) \leftarrow IY_L$ $(SP-1) \leftarrow IY_H$	・	・	・	・	・	・	・	2	15	11 111 101 11 100 101		
エク スチ エン ジ 命 令	POP qq	$qq_H \leftarrow (SP+1)$ $qq_L \leftarrow (SP)$	・	・	・	・	・	・	1	10	11 qq0 001	Ⓑ		
	POP IX	$IX_H \leftarrow (SP+1)$ $IX_L \leftarrow (SP)$	・	・	・	・	・	・	2	14	11 011 101 11 100 001			
	POP IY	$IY_H \leftarrow (SP+1)$ $IY_L \leftarrow (SP)$	・	・	・	・	・	・	2	14	11 111 101 11 100 011			
	EX DE, HL	$DE \leftrightarrow HL$	・	・	・	・	・	・	1	4	11 101 011			
	EX AF, AF'	$AF \leftrightarrow AF'$	・	・	・	・	・	・	1	4	00 001 000			
	EXX	$(BC \leftrightarrow BC')$ $(DE \leftrightarrow DE')$ $(HL \leftrightarrow HL')$	・	・	・	・	・	・	1	4	11 011 001			
ブ ロ ッ ク 転 送 命 令	EX (SP), HL	$H \leftrightarrow (SP+1)$ $L \leftrightarrow (SP)$	・	・	・	・	・	・	1	19	11 100 011			
	EX (SP), IX	$IX_H \leftrightarrow (SP+1)$ $IX_L \leftrightarrow (SP)$	・	・	・	・	・	・	2	23	11 011 101 11 100 011			
	EX (SP), IY	$IY_H \leftrightarrow (SP+1)$ $IY_L \leftrightarrow (SP)$	・	・	・	・	・	・	2	23	11 111 101 11 100 011			
	LDI	$(DE) \leftarrow (HL)$ $DE \leftarrow DE+1$ $HL \leftarrow HL+1$ $BC \leftarrow BC-1$	・	・	0	①	0	・	2	16	11 101 101 10 100 000			
	LDIR	$(DE) \leftarrow (HL)$ $DE \leftarrow DE+1$ $HL \leftarrow HL+1$ $BC \leftarrow BC-1$ until $BC=0$	・	・	0	0	0	・	2	21 if $BC \neq 0$ 16 if $BC=0$	11 101 101 10 110 000			

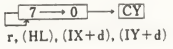
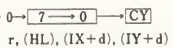
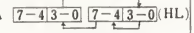
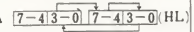
命令群	ニーモニック	オペレーション	フ ラ グ					バイト	ステート	O Pコード						
			S	Z	H	P/V	N			C	76	543	210			
ブ ロ ッ ク 転 送 命 令	LDD	(DE)←(HL) DE←DE-1 HL←HL-1 BC←BC-1	・	・	0	↑	↓	0	・	2	16	11 10	101 101	101 000		
	LDDR	(DE)←(HL) DE←DE-1 HL←HL-1 BC←BC-1 until BC=0	・	・	0	0	0	0	・	2	21 if BC≠0 16 if BC=0	11 10	101 111	101 000		
ブ ロ ッ ク ・ サ ー チ 命 令	CPI	A←(HL) HL←HL+1 BC←BC-1	↑	↑	↑	↑	↑	↓	1	・	2	16	11 10	101 100	101 001	
	CPIR	A←(HL) HL←HL+1 BC←BC-1 until A=(HL) or BC=0	↑	↑	↑	↑	↑	↑	↓	1	・	2	21 if BC≠0 and A≠(HL) 16 if BC=0 or A=(HL)	11 10	101 110	101 001
	CPD	A←(HL) HL←HL-1 BC←BC-1	↑	↑	↑	↑	↑	↑	↓	1	・	2	16	11 10	101 101	101 001
	CPDR	A←(HL) HL←HL-1 BC←BC-1 until A=(HL) or BC=0	↑	↑	↑	↑	↑	↑	↑	↓	1	・	2	21 if BC≠0 and A≠(HL) 16 if BC=0 or A=(HL)	11 10	101 111
8 ビ ッ ト 算 術 論 理 演 算 命 令	ADD A, r	A←A+r	↑	↑	↑	↑	V	0	↑	↑	1	4	10	000	r	㊦
	ADD A, n	A←A+n	↑	↑	↑	↑	V	0	↑	↑	2	7	11	000	110	← n →
	ADD A, (HL)	A←A+(HL)	↑	↑	↑	↑	V	0	↑	↑	1	7	10	000	110	
	ADD A, (IX+d)	A←A+(IX+d)	↑	↑	↑	↑	V	0	↑	↑	3	19	11	011	101	
			↑	↑	↑	↑	V	0	↑	↑			10	000	110	← d →
	ADD, A, (IY+d)	A←A+(IY+d)	↑	↑	↑	↑	V	0	↑	↑	3	19	11	111	101	
			↑	↑	↑	↑	V	0	↑	↑			10	000	110	← d →
	ADC A, r	A←A+r+CY	↑	↑	↑	↑	V	0	↑	↑	1	4	10	001	r	㊦
	ADC A, n	A←A+n+CY	↑	↑	↑	↑	V	0	↑	↑	2	7	11	001	110	← n →
	ADC A, (HL)	A←A+(HL)+CY	↑	↑	↑	↑	V	0	↑	↑	1	7	10	001	110	
	ADC A, (IX+d)	A←A+(IX+d)+CY	↑	↑	↑	↑	V	0	↑	↑	3	19	11	011	101	
			↑	↑	↑	↑	V	0	↑	↑			10	001	110	← d →
	ADC A, (IY+d)	A←A+(IY+d)+CY	↑	↑	↑	↑	V	0	↑	↑	3	19	11	111	101	
			↑	↑	↑	↑	V	0	↑	↑			10	001	110	← d →
	SUB r	A←A-r	↑	↑	↑	↑	V	1	↑	↑	↑	1	4	10	010	r
SUB n	A←A-n	↑	↑	↑	↑	V	1	↑	↑	↑	2	7	11	010	110	← n →
SUB (HL)	A←A-(HL)	↑	↑	↑	↑	V	1	↑	↑	↑	1	7	10	010	110	
SUB (IX+d)	A←A-(IX+d)	↑	↑	↑	↑	V	1	↑	↑	↑	3	19	11	011	101	
		↑	↑	↑	↑	V	1	↑	↑	↑			10	010	110	← d →
SUB (IY+d)	A←A-(IY+d)	↑	↑	↑	↑	V	1	↑	↑	↑	3	19	11	111	101	
		↑	↑	↑	↑	V	1	↑	↑	↑			10	010	110	← d →
SBC A, r	A←A-r-CY	↑	↑	↑	↑	V	1	↑	↑	↑	1	4	10	011	r	㊦
SBC A, n	A←A-n-CY	↑	↑	↑	↑	V	1	↑	↑	↑	2	7	11	011	110	← n →
SBC A, (HL)	A←A-(HL)-CY	↑	↑	↑	↑	V	1	↑	↑	↑	1	7	10	011	110	



命令群	ニーモニック	オペレーション	フ ラ グ					バイト	ステート	O P コード			
			S	Z	H	P/V	N			C	76	543	210
8 ビ ッ ト 算 術 論 理 演 算 命 令	SBC A, (IX+d)	$A \leftarrow A - (IX+d) - CY$	↑	↑	↑	V	1	↑	3	19	11 011 101 10 011 110 ← d →		
	SBC A, (IY+d)	$A \leftarrow A - (IY+d) - CY$	↑	↑	↑	V	1	↑	3	19	11 111 101 10 011 110 ← d →		
	AND r	$A \leftarrow A \wedge r$	↑	↑	1	P	0	0	1	4	10 100 r	(E)	
	AND n	$A \leftarrow A \wedge n$	↑	↑	1	P	0	0	2	7	11 100 110 ← n →		
	AND (HL)	$A \leftarrow A \wedge (HL)$	↑	↑	1	P	0	0	1	7	10 100 110		
	AND (IX+d)	$A \leftarrow A \wedge (IX+d)$	↑	↑	1	P	0	0	3	19	11 011 101 10 100 110 ← d →		
	AND (IY+d)	$A \leftarrow A \wedge (IY+d)$	↑	↑	1	P	0	0	3	19	11 111 101 10 100 110 ← d →		
	OR r	$A \leftarrow A \vee r$	↑	↑	0	P	0	0	1	4	10 110 r	(E)	
	OR n	$A \leftarrow A \vee n$	↑	↑	0	P	0	0	2	7	11 110 110 ← n →		
	OR (HL)	$A \leftarrow A \vee (HL)$	↑	↑	0	P	0	0	1	7	10 110 110		
	OR (IX+d)	$A \leftarrow A \vee (IX+d)$	↑	↑	0	P	0	0	3	19	11 011 101 10 110 110 ← d →		
	OR (IY+d)	$A \leftarrow A \vee (IY+d)$	↑	↑	0	P	0	0	3	19	11 111 101 10 110 110 ← d →		
	XOR r	$A \leftarrow A \oplus r$	↑	↑	0	P	0	0	1	4	10 101 r	(E)	
	XOR n	$A \leftarrow A \oplus n$	↑	↑	0	P	0	0	2	7	11 101 110 ← n →		
	XOR (HL)	$A \leftarrow A \oplus (HL)$	↑	↑	0	P	0	0	1	7	10 101 110		
	XOR (IX+d)	$A \leftarrow A \oplus (IX+d)$	↑	↑	0	P	0	0	3	19	11 011 101 10 101 110 ← d →		
	XOR (IY+d)	$A \leftarrow A \oplus (IY+d)$	↑	↑	0	P	0	0	3	19	11 111 101 10 101 110 ← d →		
	CP r	$A - r$	↑	↑	↑	V	1	↑	1	4	10 111 r	(E)	
	CP n	$A - n$	↑	↑	↑	V	1	↑	2	7	11 111 110 ← n →		
	CP (HL)	$A - (HL)$	↑	↑	↑	V	1	↑	1	7	10 111 110		
	CP (IX+d)	$A - (IX+d)$	↑	↑	↑	V	1	↑	3	19	11 011 101 10 111 110 ← d →		
	CP (IY+d)	$A - (IY+d)$	↑	↑	↑	V	1	↑	3	19	11 111 101 10 111 110 ← d →		
	INC r	$r \leftarrow r + 1$	↑	↑	↑	V	0	•	1	4	00 r 100	(E)	
	INC (HL)	$(HL) \leftarrow (HL) + 1$	↑	↑	↑	V	0	•	1	11	00 110 100		
	INC (IX+d)	$(IX+d) \leftarrow (IX+d) + 1$	↑	↑	↑	V	0	•	3	23	11 011 101 00 110 100 ← d →		
	INC (IY+d)	$(IY+d) \leftarrow (IX+d) + 1$	↑	↑	↑	V	0	•	3	23	11 111 101 00 110 100 ← d →		
	DEC r	$r \leftarrow r - 1$	↑	↑	↑	V	1	•	1	4	00 r 101	(E)	
	DEC (HL)	$(HL) \leftarrow (HL) - 1$	↑	↑	↑	V	1	•	1	11	00 110 101		
	DEC (IX+d)	$(IX+d) \leftarrow (IX+d) - 1$	↑	↑	↑	V	1	•	3	23	11 011 101 00 110 101 ← d →		

命令群	ニーモニック	オペレーション	フ ラ グ					バイト	ステート	O P コード		
			S	Z	H	P/V	N	C		76	543	210
演算8ビット 論理命令	DEC (IY+d)	$(IY+d) \leftarrow (IY+d) - 1$	↑	↑	↑	V	1	・	3	23	11 111 101 00 110 101 ← d →	
16 ビ ット 算 術 演 算 命 令	ADD HL, ss	$HL \leftarrow HL + ss$	・	・	×	・	0	↑	1	11	00 ss1 001	Ⓐ
	ADC HL, ss	$HL \leftarrow HL + ss + CY$	↑	↑	×	V	0	↑	2	15	11 101 101 01 ss1 010	Ⓐ
	SBC HL, ss	$HL \leftarrow HL - ss - CY$	↑	↑	×	V	1	↑	2	15	11 101 101 01 ss0 010	Ⓐ
	ADD IX, pp	$IX \leftarrow IX + pp$	・	・	×	・	0	↑	2	15	11 011 101 00 ppl 001	Ⓒ
	ADD IY, rr	$IY \leftarrow IY + rr$	・	・	×	・	0	↑	2	15	11 111 101 00 rrl 001	Ⓓ
	INC ss	$ss \leftarrow ss + 1$	・	・	・	・	・	・	1	6	00 ss0 011	Ⓐ
	INC IX	$IX \leftarrow IX + 1$	・	・	・	・	・	・	2	10	11 011 101 00 100 011	
	INC IY	$IY \leftarrow IY + 1$	・	・	・	・	・	・	2	10	11 111 101 00 100 011	
	DEC ss	$ss \leftarrow ss - 1$	・	・	・	・	・	・	1	6	00 ss1 011	Ⓐ
	DEC IX	$IX \leftarrow IX - 1$	・	・	・	・	・	・	2	10	11 011 101 00 101 011	
	DEC IY	$IY \leftarrow IY - 1$	・	・	・	・	・	・	2	10	11 111 101 00 101 011	
ア キ ュ ム レ ー タ 操 作 命 令	DAA	Decimal adjust Acc	↑	↑	↑	P	・	↑	1	4	00 100 111	
	CPL	$A \leftarrow \bar{A}$	・	・	1	・	1	・	1	4	00 101 111	
	NEG	$A \leftarrow \bar{A} + 1$	↑	↑	↑	V	1	↑	2	8	11 101 101 01 000 100	
	CCF	$CY \leftarrow \bar{CY}$	・	・	×	・	0	↑	1	4	00 111 111	
	SCF	$CY \leftarrow 1$	・	・	0	・	0	1	1	4	00 110 111	
C P U コ ン ト ロ ー ル 命 令	NOP	No operation	・	・	・	・	・	・	1	4	00 000 000	
	HALT	CPU halted	・	・	・	・	・	・	1	4	01 110 110	
	DI	$IFF \leftarrow 0$	・	・	・	・	・	・	1	4	11 110 011	
	EI	$IFF \leftarrow 1$	・	・	・	・	・	・	1	4	11 111 011	
	IM 0	Set interrupt mode 0	・	・	・	・	・	・	2	8	11 101 101 01 000 110	
	IM 1	Set interrupt mode 1	・	・	・	・	・	・	2	8	11 101 101 01 010 110	
	IM 2	Set interrupt mode 2	・	・	・	・	・	・	2	8	11 101 101 01 011 110	
ロ ー テ ー ト ・ シ フ ト 命 令	RLCA		・	・	0	・	0	↑	1	4	00 000 111	
	RLA		・	・	0	・	0	↑	1	4	00 010 111	
	RRCA		・	・	0	・	0	↑	1	4	00 001 111	
	RRA		・	・	0	・	0	↑	1	4	00 011 111	

命令群	ニーモニック	オペレーション	フ ラ グ					バイト	ステート	OPコード		
			S	Z	H	P/V	N			76	543	210
ロ テ ィ ト ・ シ フ ト 命 令	RLC r		↑	↑	0	P	0	↑	2	8	11 001 011	00 000 r (E)
	RLC (HL)		↑	↑	0	P	0	↑	2	15	11 001 011	00 000 110
	RLC (IX+d)	 r, (HL), (IX+d), (IY+d)	↑	↑	0	P	0	↑	4	23	11 011 101	11 001 011 ← d → 00 000 110
	RLC (IY+d)		↑	↑	0	P	0	↑	4	23	11 111 101	11 001 011 ← d → 00 000 110
	RL r		↑	↑	0	P	0	↑	2	8	11 001 011	00 010 r (E)
	RL (HL)		↑	↑	0	P	0	↑	2	15	11 001 011	00 010 110
	RL (IX+d)	 r, (HL), (IX+d), (IY+d)	↑	↑	0	P	0	↑	4	23	11 011 101	11 001 011 ← d → 00 010 110
	RL (IY+d)		↑	↑	0	P	0	↑	4	23	11 111 101	11 001 011 ← d → 00 010 110
	RRC r		↑	↑	0	P	0	↑	2	8	11 001 011	00 001 r (E)
	RRC (HL)		↑	↑	0	P	0	↑	2	15	11 001 011	00 001 110
	RRC (IX+d)	 r, (HL), (IX+d), (IY+d)	↑	↑	0	P	0	↑	4	23	11 011 101	11 001 011 ← d → 00 001 110
	RRC (IY+d)		↑	↑	0	P	0	↑	4	23	11 111 101	11 001 011 ← d → 00 001 110
	RR r		↑	↑	0	P	0	↑	2	8	11 001 011	00 011 r (E)
	RR (HL)		↑	↑	0	P	0	↑	2	15	11 001 011	00 011 110
	RR (IX+d)	 r, (HL), (IX+d), (IY+d)	↑	↑	0	P	0	↑	4	23	11 011 101	11 001 011 ← d → 00 011 110
	RR (IY+d)		↑	↑	0	P	0	↑	4	23	11 111 101	11 001 011 ← d → 00 011 110
	SLA r		↑	↑	0	P	0	↑	2	8	11 001 011	00 100 r (E)
	SLA (HL)		↑	↑	0	P	0	↑	2	15	11 001 011	00 100 110
	SLA (IX+d)	 r, (HL), (IX+d), (IY+d)	↑	↑	0	P	0	↑	4	23	11 011 101	11 001 011 ← d → 00 100 110
	SLA (IY+d)		↑	↑	0	P	0	↑	4	23	11 111 101	11 001 011 ← d → 00 100 110

命令群	ニーモニック	オペレーション	フ ラ グ					バイト	ステート	O P コード		
			S	Z	H	P/V	N			76	543	210
ロー レ ジ ス タ ・ シ フ ト 命 令	SRA r		↑	↑	0	P	0	↑	2	8	11 001 011 00 101 r	Ⓔ
	SRA (HL)		↑	↑	0	P	0	↑	2	15	11 001 011 00 101 110	
	SRA (IX+d)		↑	↑	0	P	0	↑	4	23	11 011 101 11 001 011 ← d → 00 101 110	
	SRA (IY+d)		↑	↑	0	P	0	↑	4	23	11 111 101 11 001 011 ← d → 00 101 110	
	SRL r		↑	↑	0	P	0	↑	2	8	11 001 011 00 111 r	Ⓔ
	SRL (HL)		↑	↑	0	P	0	↑	2	15	11 001 011 00 111 110	
	SRL (IX+d)		↑	↑	0	P	0	↑	4	23	11 011 101 11 001 011 ← d → 00 111 110	
	SRL (IY+d)		↑	↑	0	P	0	↑	4	23	11 111 101 11 001 011 ← d → 00 111 110	
	RLD	A 	↑	↑	0	P	0	↑	2	18	11 101 101 01 101 111	
	RRD	A 	↑	↑	0	P	0	↑	2	18	11 101 101 01 100 111	
ビ ッ ト 操 作 命 令	BIT b, r	Z ← r <sub>b</sub>	×	↑	1	×	0	↑	2	8	11 001 011 01 b r	Ⓔ Ⓕ
	BIT b, (HL)	Z ← (HL) <sub>b</sub>	×	↑	1	×	0	↑	2	12	11 001 011 01 b 110	Ⓕ
	BIT b, (IX+d)	Z ← (IX+d) <sub>b</sub>	×	↑	1	×	0	↑	4	20	11 011 101 11 001 011 ← d → 01 b 110	Ⓕ
	BIT b, (IY+d)	Z ← (IY+d) <sub>b</sub>	×	↑	1	×	0	↑	4	20	11 111 101 11 001 011 ← d → 01 b 110	Ⓕ
	SET b, r	r <sub>b</sub> ← 1	•	•	•	•	•	•	2	8	11 001 011 11 b r	Ⓔ Ⓕ
	SET b, (HL)	(HL) <sub>b</sub> ← 1	•	•	•	•	•	•	2	15	11 001 011 11 b 110	Ⓕ
	SET b, (IX+d)	(IX+d) <sub>b</sub> ← 1	•	•	•	•	•	•	4	23	11 011 101 11 001 011 ← d → 11 b 110	Ⓕ
	SET b, (IY+d)	(IY+d) <sub>b</sub> ← 1	•	•	•	•	•	•	4	23	11 111 101 11 001 011 ← d → 11 b 110	Ⓕ
	RES b, r	r <sub>b</sub> ← 0	•	•	•	•	•	•	2	8	11 001 011 10 b r	Ⓔ Ⓕ
	RES b, (HL)	(HL) <sub>b</sub> ← 0	•	•	•	•	•	•	2	15	11 001 011 10 b 110	Ⓕ
	RES b, (IX+d)	(IX+d) <sub>b</sub> ← 0	•	•	•	•	•	•	4	23	11 011 101 11 001 011 ← d → 10 b 110	Ⓕ
	RES b, (IY+d)	(IY+d) <sub>b</sub> ← 0	•	•	•	•	•	•	4	23	11 111 101 11 001 011 ← d → 10 b 110	Ⓕ

命令群	ニーモニック	オペレーション	フ ラ グ					バイト	ステート	OPコード		
			S	Z	H	P/V	N			C	76	543
ジャンプ・コール命令	JP nn	PC←nn	•	•	•	•	•	•	3	10	11 000 011 ← n → ← n →	
	JP cc, nn	If cc is true PC←nn Otherwise continue	•	•	•	•	•	•	3	10	11 cc 010 ⑥ ← n → ← n →	
	JR e	PC←PC+e	•	•	•	•	•	•	2	12	00 011 000 ← e-2 →	
	JR C, e	If C=0 continue If C=1 PC←PC+e	•	•	•	•	•	•	2	7 if C=0 12 if C=1	00 111 000 ← e-2 →	
	JR NC, e	If C=1 continue If C=0 PC←PC+e	•	•	•	•	•	•	2	7 if C=1 12 if C=0	00 110 000 ← e-2 →	
	JR Z, e	If Z=0 continue If Z=1 PC←PC+e	•	•	•	•	•	•	2	7 if Z=0 12 if Z=1	00 101 000 ← e-2 →	
	JR NZ, e	If Z=1 continue If Z=0 PC←PC+e	•	•	•	•	•	•	2	7 if Z=1 12 if Z=0	00 100 000 ← e-2 →	
	JP (HL)	PC←HL	•	•	•	•	•	•	1	4	11 101 001	
	JP (IX)	PC←IX	•	•	•	•	•	•	2	8	11 011 101 11 101 001	
	JP (IY)	PC←IY	•	•	•	•	•	•	2	8	11 111 101 11 101 001	
	DJNZ e	B←B-1 if B=0 continue if B≠0 PC←PC+e	•	•	•	•	•	•	2	8 if B=0 13 if B≠0	00 010 000 ← e-2 →	
	CALL nn	(SP-1)←PC <sub>H</sub> (SP-2)←PC <sub>L</sub> PC←nn	•	•	•	•	•	•	3	17	11 001 101 ← n → ← n →	
	CALL cc, nn	If cc is false continue otherwise same as CALL nn	•	•	•	•	•	•	3	10 if cc is false 17 if cc is true	11 cc 100 ⑥ ← n → ← n →	
	RET	PC <sub>L</sub> ←(SP) PC <sub>H</sub> ←(SP+1)	•	•	•	•	•	•	1	10	11 001 001	
	RET cc	If cc is false continue otherwise same as RET	•	•	•	•	•	•	1	5 if cc is false 11 if cc is true	11 cc 000 ⑥	
RETI	Return from interrupt	•	•	•	•	•	•	2	14	11 101 101 01 001 101		
RETN	Return from non maskable interrupt	•	•	•	•	•	•	2	14	11 101 101 01 000 101		
RST p	(SP-1)←PC <sub>H</sub> (SP-2)←PC <sub>L</sub> PC <sub>H</sub> ←0 PC <sub>L</sub> ←p	•	•	•	•	•	•	1	11	11 t 111 ⑩		
入出力命令	IN A, n	A←(n) A <sub>0-7</sub> ←n A <sub>8-15</sub> ←A	•	•	•	•	•	•	2	11	11 011 011 ← n →	
	IN r, (C)	r←(C) if r=110 only the flags will be affected A <sub>0-7</sub> ←C A <sub>8-15</sub> ←B	↑	↑	↑	P	0	•	2	12	11 101 101 01 r 000 ⑬	
	INI	(HL)←(C) B←B-1 HL←HL+1 A <sub>0-7</sub> ←C A <sub>8-15</sub> ←B	×	③ ↑	×	×	1	•	2	16	11 101 101 10 100 010	
	INIR	(HL)←(C) B←B-1 HL←HL+1 until B=0 A <sub>0-7</sub> ←C A <sub>8-15</sub> ←B	×	1	×	×	1	•	2	21 if B≠0 16 if B=0	11 101 101 10 110 010	
	IND	(HL)←(C) B←B-1 HL←HL-1 A <sub>0-7</sub> ←C A <sub>8-15</sub> ←B	×	③ ↑	×	×	1	•	2	16	11 101 101 10 101 010	

命令群	ニーモニック	オペレーション	フ ラ グ					バイト	ステート	O P コード			
			S	Z	H	P/V	N			C	76	543	210
入 出 力 命 令	INDR	(HL) ← (C) B ← B-1 HL ← HL-1 until B=0 A <sub>0-7</sub> ← C A <sub>8-15</sub> ← B	×	1	×	×	1	•	2	21 if B=0 16 if B=0	11 10	101 111	101 010
	OUT n, A	(n) ← A A <sub>0-7</sub> ← n A <sub>8-15</sub> ← A	•	•	•	•	•	•	2	11	11 ←	010 n	011 →
	OUT (C), r	(C) ← r A <sub>0-7</sub> ← C A <sub>8-15</sub> ← B	•	•	•	•	•	•	2	12	11 01	101 r	101 001 ⑤
	OUTI	(C) ← (HL) B ← B-1 HL ← HL+1 A <sub>0-7</sub> ← C A <sub>8-15</sub> ← B	×	③ ↓	×	×	1	•	2	16	11 10	101 100	101 011
	OTIR	(C) ← (HL) B ← B-1 HL ← HL+1 until B=0 A <sub>0-7</sub> ← C A <sub>8-15</sub> ← B	×	1	×	×	1	•	2	21 if B=0 16 if B=0	11 10	101 110	101 011
	OUTD	(C) ← (HL) B ← B-1 HL ← HL-1 A <sub>0-7</sub> ← C A <sub>8-15</sub> ← B	×	③ ↓	×	×	1	•	2	16	11 10	101 101	101 011
	OTDR	(C) ← (HL) B ← B-1 HL ← HL-1 until B=0 A <sub>0-7</sub> ← C A <sub>8-15</sub> ← B	×	1	×	×	1	•	2	21 if B=0 16 if B=0	11 10	101 111	101 011

A		B		C		D		E		F		G			H	
Reg	ss dd	Reg	qq	Reg	pp	Reg	rr	Reg	r, r'	Bit	b	cc	Condition	Flag	P	t
BC	00	BC	00	BC	00	BC	00	B	000	0	000	000	NZ Non Zero	Z	00H	000
DE	01	DE	01	DE	01	DE	01	C	001	1	001	001	Z Zero	Z	08H	001
HL	10	HL	10	IX	10	IX	10	D	010	2	010	010	NC Non Carry	C	10H	010
SP	11	AF	11	SP	11	SP	11	E	011	3	011	011	C Carry	C	18H	011
								H	100	4	100	100	PO Parity Odd	P/V	20H	100
								L	101	5	101	101	PE Parity Even	P/V	28H	101
								A	111	6	110	110	P Sign Positive	S	30H	110
										7	111	111	M Sign Negative	S	38H	111

d, n : 8ビット・イミューディエト・データ

e : 相対アドレッシングの変位置

e-2 : eの実効変位置

#### フラグ

- : 影響受けない
- ↑ : 演算結果に従った影響を受ける
- ① BC-1=0 ならば P/V=0, その他 P/V=1
- 0 : リセット
- P : "1" 偶数パリティ, "0" 奇数パリティ
- ② A=(HL) ならば Z=1, その他 Z=0
- 1 : セット
- V : "1" オーバフロー有り, "0" オーバフロー無し
- ③ B-1=0 ならば Z=1, その他 Z=0
- ×
- IFF : P/Vフラグ ← (IFF)

# 2. $\mu$ COM-82 機械語 $\leftrightarrow$ ニーモニック対応表

機 械 語  $\longleftrightarrow$  ニーモニック

00 NOP	40 LD B, B	80 ADD A, B	C0 RET NZ
01 LD BC, nn	41 LD B, C	81 ADD A, C	C1 POP BC
02 LD (BC), A	42 LD B, D	82 ADD A, D	C2 JP NZ, nn
03 INC BC	43 LD B, E	83 ADD A, E	C3 JP nn
04 INC B	44 LD B, H	84 ADD A, H	C4 CALL NZ, nn
05 DEC B	45 LD B, L	85 ADD A, L	C5 PUSH BC
06 LD B, n	46 LD B, (HL)	86 ADD A, (HL)	C6 ADD A, n
07 RLCA	47 LD B, A	87 ADD A, A	C7 RST 00H
08 EX AF, AF	48 LD C, B	88 ADC A, B	C8 RET Z
09 ADD HL, BC	49 LD C, C	89 ADC A, C	C9 RET
0A LD A, (BC)	4A LD C, D	8A ADC A, D	CA JP Z, nn
0B DEC BC	4B LD C, E	8B ADC A, E	CB <span style="border: 1px solid black; display: inline-block; width: 50px; height: 1em; vertical-align: middle;"></span>
0C INC C	4C LD C, H	8C ADC A, H	CC CALL Z, nn
0D DEC C	4D LD C, L	8D ADC A, L	CD CALL nn
0E LD C, n	4E LD C, (HL)	8E ADC A, (HL)	CE ADC A, n
0F RRCA	4F LD C, A	8F ADC A, A	CF RST 08H
10 DJNZ e	50 LD D, B	90 SUB B	D0 RET NC
11 LD DE, nn	51 LD D, C	91 SUB C	D1 POP DE
12 LD (DE), A	52 LD D, D	92 SUB D	D2 JP NC, nn
13 INC DE	53 LD D, E	93 SUB E	D3 OUT n, A
14 INC D	54 LD D, H	94 SUB H	D4 CALL NC, nn
15 DEC D	55 LD D, L	95 SUB L	D5 PUSH DE
16 LD D, n	56 LD D, (HL)	96 SUB (HL)	D6 SUB n
17 RL A	57 LD D, A	97 SUB A	D7 RST 10H
18 JR e	58 LD E, B	98 SBC A, B	D8 RET C
19 ADD HL, DE	59 LD E, C	99 SBC A, C	D9 EXX
1A LD A, (DE)	5A LD E, D	9A SBC A, D	DA JP C, nn
1B DEC DE	5B LD E, E	9B SBC A, E	DB IN A, n
1C INC E	5C LD E, H	9C SBC A, H	DC CALL C, nn
1D DEC E	5D LD E, L	9D SBC A, L	DD <span style="border: 1px solid black; display: inline-block; width: 50px; height: 1em; vertical-align: middle;"></span>
1E LD E, n	5E LD E, (HL)	9E SBC A, (HL)	DE SBC A, n
1F RRA	5F LD E, A	9F SBC A, A	DF RST 18H
20 JR NZ, e	60 LD H, B	A0 AND B	E0 RET PO
21 LD HL, nn	61 LD H, C	A1 AND C	E1 POP HL
22 LD (nn), HL	62 LD H, D	A2 AND D	E2 JP PO, nn
23 INC HL	63 LD H, E	A3 AND E	E3 EX (SP), HL
24 INC H	64 LD H, H	A4 AND H	E4 CALL PO, nn
25 DEC H	65 LD H, L	A5 AND L	E5 PUSH HL
26 LD H, n	66 LD H, (HL)	A6 AND (HL)	E6 AND n
27 DAA	67 LD H, A	A7 AND A	E7 RST 20H
28 JR Z, e	68 LD L, B	A8 XOR B	E8 RET PE
29 ADD HL, HL	69 LD L, C	A9 XOR C	E9 JP (HL)
2A LD HL, (nn)	6A LD L, D	AA XOR D	EA JP PE, nn
2B DEC HL	6B LD L, E	AB XOR E	EB EX DE, HL
2C INC L	6C LD L, H	AC XOR H	EC CALL PE, nn
2D DEC L	6D LD L, L	AD XOR L	ED <span style="border: 1px solid black; display: inline-block; width: 50px; height: 1em; vertical-align: middle;"></span>
2E LD L, n	6E LD L, (HL)	AE XOR (HL)	EE XOR n
2F CPL	6F LD L, A	AF XOR A	EF RST 28H
30 JR NC, e	70 LD (HL), B	B0 OR B	F0 RET P
31 LD SP, nn	71 LD (HL), C	B1 OR C	F1 POP AF
32 LD (nn), A	72 LD (HL), D	B2 OR D	F2 JP P, nn
33 INC SP	73 LD (HL), E	B3 OR E	F3 DI
34 INC (HL)	74 LD (HL), H	B4 OR H	F4 CALL P, nn
35 DEC (HL)	75 LD (HL), L	B5 OR L	F5 PUSH AF
36 LD (HL), n	76 HALT	B6 OR (HL)	F6 OR n
37 SCF	77 LD (HL), A	B7 OR A	F7 RST 30H
38 JR C, e	78 LD A, B	B8 CP B	F8 RET M
39 ADD HL, SP	79 LD A, C	B9 CP C	F9 LD SP, HL
3A LD A, (nn)	7A LD A, D	BA CP D	FA JP M, nn
3B DEC SP	7B LD A, E	BB CP E	FB EI
3C INC A	7C LD A, H	BC CP H	FC CALL M, nn
3D DEC A	7D LD A, L	BD CP L	FD <span style="border: 1px solid black; display: inline-block; width: 50px; height: 1em; vertical-align: middle;"></span>
3E LD A, n	7E LD A, (HL)	BE CP (HL)	FE CP n
3F CCF	7F LD A, A	BF CP A	FF RST 38H



## C B ××

00 RLC B	40 BIT 0,B	80 RES 0,B	C0 SET 0,B
01 RLC C	41 BIT 0,C	81 RES 0,C	C1 SET 0,C
02 RLC D	42 BIT 0,D	82 RES 0,D	C2 SET 0,D
03 RLC E	43 BIT 0,E	83 RES 0,E	C3 SET 0,E
04 RLC H	44 BIT 0,H	84 RES 0,H	C4 SET 0,H
05 RLC L	45 BIT 0,L	85 RES 0,L	C5 SET 0,L
06 RLC (HL)	46 BIT 0,(HL)	86 RES 0,(HL)	C6 SET 0,(HL)
07 RLC A	47 BIT 0,A	87 RES 0,A	C7 SET 0,A
08 RRC B	48 BIT 1,B	88 RES 1,B	C8 SET 1,B
09 RRC C	49 BIT 1,C	89 RES 1,C	C9 SET 1,C
0A RRC D	4A BIT 1,D	8A RES 1,D	CA SET 1,D
0B RRC E	4B BIT 1,E	8B RES 1,E	CB SET 1,E
0C RRC H	4C BIT 1,H	8C RES 1,H	CC SET 1,H
0D RRC L	4D BIT 1,L	8D RES 1,L	CD SET 1,L
0E RRC (HL)	4E BIT 1,(HL)	8E RES 1,(HL)	CE SET 1,(HL)
0F RRC A	4F BIT 1,A	8F RES 1,A	CF SET 1,A
10 RL B	50 BIT 2,B	90 RES 2,B	D0 SET 2,B
11 RL C	51 BIT 2,C	91 RES 2,C	D1 SET 2,C
12 RL D	52 BIT 2,D	92 RES 2,D	D2 SET 2,D
13 RL E	53 BIT 2,E	93 RES 2,E	D3 SET 2,E
14 RL H	54 BIT 2,H	94 RES 2,H	D4 SET 2,H
15 RL L	55 BIT 2,L	95 RES 2,L	D5 SET 2,L
16 RL (HL)	56 BIT 2,(HL)	96 RES 2,(HL)	D6 SET 2,(HL)
17 RL A	57 BIT 2,A	97 RES 2,A	D7 SET 2,A
18 RR B	58 BIT 3,B	98 RES 3,B	D8 SET 3,B
19 RR C	59 BIT 3,C	99 RES 3,C	D9 SET 3,C
1A RR D	5A BIT 3,D	9A RES 3,D	DA SET 3,D
1B RR E	5B BIT 3,E	9B RES 3,E	DB SET 3,E
1C RR H	5C BIT 3,H	9C RES 3,H	DC SET 3,H
1D RR L	5D BIT 3,L	9D RES 3,L	DD SET 3,L
1E RR (HL)	5E BIT 3,(HL)	9E RES 3,(HL)	DE SET 3,(HL)
1F RR A	5F BIT 3,A	9F RES 3,A	DF SET 3,A
20 SLA B	60 BIT 4,B	A0 RES 4,B	E0 SET 4,B
21 SLA C	61 BIT 4,C	A1 RES 4,C	E1 SET 4,C
22 SLA D	62 BIT 4,D	A2 RES 4,D	E2 SET 4,D
23 SLA E	63 BIT 4,E	A3 RES 4,E	E3 SET 4,E
24 SLA H	64 BIT 4,H	A4 RES 4,H	E4 SET 4,H
25 SLA L	65 BIT 4,L	A5 RES 4,L	E5 SET 4,L
26 SLA (HL)	66 BIT 4,(HL)	A6 RES 4,(HL)	E6 SET 4,(HL)
27 SLA A	67 BIT 4,A	A7 RES 4,A	E7 SET 4,A
28 SRA B	68 BIT 5,B	A8 RES 5,B	E8 SET 5,B
29 SRA C	69 BIT 5,C	A9 RES 5,C	E9 SET 5,C
2A SRA D	6A BIT 5,D	AA RES 5,D	EA SET 5,D
2B SRA E	6B BIT 5,E	AB RES 5,E	EB SET 5,E
2C SRA H	6C BIT 5,H	AC RES 5,H	EC SET 5,H
2D SRA L	6D BIT 5,L	AD RES 5,L	ED SET 5,L
2E SRA (HL)	6E BIT 5,(HL)	AE RES 5,(HL)	EE SET 5,(HL)
2F SRA A	6F BIT 5,A	AF RES 5,A	EF SET 5,A
30	70 BIT 6,B	B0 RES 6,B	F0 SET 6,B
31	71 BIT 6,C	B1 RES 6,C	F1 SET 6,C
32	72 BIT 6,D	B2 RES 6,D	F2 SET 6,D
33	73 BIT 6,E	B3 RES 6,E	F3 SET 6,E
34	74 BIT 6,H	B4 RES 6,H	F4 SET 6,H
35	75 BIT 6,L	B5 RES 6,L	F5 SET 6,L
36	76 BIT 6,(HL)	B6 RES 6,(HL)	F6 SET 6,(HL)
37	77 BIT 6,A	B7 RES 6,A	F7 SET 6,A
38 SRL B	78 BIT 7,B	B8 RES 7,B	F8 SET 7,B
39 SRL C	79 BIT 7,C	B9 RES 7,C	F9 SET 7,C
3A SRL D	7A BIT 7,D	BA RES 7,D	FA SET 7,D
3B SRL E	7B BIT 7,E	BB RES 7,E	FB SET 7,E
3C SRL H	7C BIT 7,H	BC RES 7,H	FC SET 7,H
3D SRL L	7D BIT 7,L	BD RES 7,L	FD SET 7,L
3E SRL (HL)	7E BIT 7,(HL)	BE RES 7,(HL)	FE SET 7,(HL)
3F SRL A	7F BIT 7,A	BF RES 7,A	FF SET 7,A

D D × ×			E D × ×			F D × ×		
0 9	ADD	IX, BC	4 0	IN	B, (C)	0 9	ADD	IV, BC
1 9	ADD	IX, DE	4 1	OUT	(C), B	1 9.	ADD	IV, DE
2 1	LD	IX, nn	4 2	SBC	HL, BC	2 1	LD	IV, nn
2 2	LD	(nn), IX	4 3	LD	(nn), BC	2 2	LD	(nn), IV
2 3	INC	IX	4 4	NEG		2 3	INC	IV
2 9	ADD	IX, IX	4 5	RET N		2 9	ADD	IV, IV
2 A	LD	IX, (nn)	4 6	IM	0	2 A	LD	IV, (nn)
2 B	DEC	IX	4 7	LD	I, A	2 B	DEC	IV
3 4	INC	(IX+d)	4 8	IN	C, (C)	3 4	INC	(IV+d)
3 5	DEC	(IX+d)	4 9	OUT	(C), C	3 5	DEC	(IV+d)
3 6	LD	(IX+d), n	4 A	ADC	HL, BC	3 6	LD	(IV+d), n
3 9	ADD	IX, SP	4 B	LD	BC, (nn)	3 9	ADD	IV, SP
4 6	LD	B, (IX+d)	4 D	RET I		4 6	LD	B, (IV+d)
4 E	LD	C, (IX+d)	4 F	LD	R, A	4 E	LD	C, (IV+d)
5 6	LD	D, (IX+d)	5 0	IN	D, (C)	5 6	LD	D, (IV+d)
5 E	LD	E, (IX+d)	5 1	OUT	(C), D	5 E	LD	E, (IV+d)
6 6	LD	H, (IX+d)	5 2	SBC	HL, DE	6 6	LD	H, (IV+d)
6 E	LD	L, (IX+d)	5 3	LD	(nn), DE	6 E	LD	L, (IV+d)
7 0	LD	(IX+d), B	5 6	IM	1	7 0	LD	(IV+d), B
7 1	LD	(IX+d), C	5 7	LD	A, I	7 1	LD	(IV+d), C
7 2	LD	(IX+d), D	5 8	IN	E, (C)	7 2	LD	(IV+d), D
7 3	LD	(IX+d), E	5 9	OUT	(C), E	7 3	LD	(IV+d), E
7 4	LD	(IX+d), H	5 A	ADC	HL, DE	7 4	LD	(IV+d), H
7 5	LD	(IX+d), L	5 B	LD	DE, (nn)	7 5	LD	(IV+d), L
7 7	LD	(IX+d), A	5 E	IM	2	7 7	LD	(IV+d), A
7 E	LD	A, (IX+d)	5 F	LD	A, R	7 E	LD	A, (IV+d)
8 6	ADD	A, (IX+d)	6 0	IN	H, (C)	8 6	ADD	A, (IV+d)
8 E	ADC	A, (IX+d)	6 1	OUT	(C), H	8 E	ADC	A, (IV+d)
9 6	SUB	(IX+d)	6 2	SBC	HL, HL	9 6	SUB	(IV+d)
9 E	SBC	A, (IX+d)	6 7	R R D		9 E	SBC	A, (IV+d)
A 6	AND	(IX+d)	6 8	IN	L, (C)	A 6	AND	(IV+d)
A E	XOR	(IX+d)	6 9	OUT	(C), L	A E	XOR	(IV+d)
B 6	OR	(IX+d)	6 A	ADC	HL, HL	B 6	OR	(IV+d)
B E	CP	(IX+d)	6 F	R L D		B E	CP	(IV+d)
CB d 0 6	R L C	(IX+d)	7 2	SBC	HL, SP	CB d 0 6	R L C	(IV+d)
CB d 0 E	R R C	(IX+d)	7 3	LD	(nn), SP	CB d 0 E	R R C	(IV+d)
CB d 1 6	R L	(IX+d)	7 8	IN	A, (C)	CB d 1 6	R L	(IV+d)
CB d 1 E	R R	(IX+d)	7 9	OUT	(C), A	CB d 1 E	R R	(IV+d)
CB d 2 6	S L A	(IX+d)	7 A	ADC	HL, SP	CB d 2 6	S L A	(IV+d)
CB d 2 E	S R A	(IX+d)	7 B	LD	SP, (nn)	CB d 2 E	S R A	(IV+d)
CB d 3 E	S R L	(IX+d)	A 0	L D I		CB d 3 E	S R L	(IV+d)
CB d 4 6	B I T	0, (IX+d)	A 1	C P I		CB d 4 6	B I T	0, (IV+d)
CB d 4 E	B I T	1, (IX+d)	A 2	I N I		CB d 4 E	B I T	1, (IV+d)
CB d 5 6	B I T	2, (IX+d)	A 3	O U T I		CB d 5 6	B I T	2, (IV+d)
CB d 5 E	B I T	3, (IX+d)	A 8	L D D		CB d 5 E	B I T	3, (IV+d)
CB d 6 6	B I T	4, (IX+d)	A 9	C P D		CB d 6 6	B I T	4, (IV+d)
CB d 6 E	B I T	5, (IX+d)	A A	I N D		CB d 6 E	B I T	5, (IV+d)
CB d 7 6	B I T	6, (IX+d)	A B	O U T D		CB d 7 6	B I T	6, (IV+d)
CB d 7 E	B I T	7, (IX+d)	B 0	L D I R		CB d 7 E	B I T	7, (IV+d)
CB d 8 6	R E S	0, (IX+d)	B 1	C P I R		CB d 8 6	R E S	0, (IV+d)
CB d 8 E	R E S	1, (IX+d)	B 2	I N I R		CB d 8 E	R E S	1, (IV+d)
CB d 9 6	R E S	2, (IX+d)	B 3	O T I R		CB d 9 6	R E S	2, (IV+d)
CB d 9 E	R E S	3, (IX+d)	B 8	L D D R		CB d 9 E	R E S	3, (IV+d)
CB d A 6	R E S	4, (IX+d)	B 9	C P D R		CB d A 6	R E S	4, (IV+d)
CB d A E	R E S	5, (IX+d)	B A	I N D R		CB d A E	R E S	5, (IV+d)
CB d B 6	R E S	6, (IX+d)	B B	O T D R		CB d B 6	R E S	6, (IV+d)
CB d B E	R E S	7, (IX+d)				CB d B E	R E S	7, (IV+d)
CB d C 6	S E T	0, (IX+d)				CB d C 6	S E T	0, (IV+d)
CB d C E	S E T	1, (IX+d)				CB d C E	S E T	1, (IV+d)
CB d D 6	S E T	2, (IX+d)				CB d D 6	S E T	2, (IV+d)
CB d D E	S E T	3, (IX+d)				CB d D E	S E T	3, (IV+d)
CB d E 6	S E T	4, (IX+d)				CB d E 6	S E T	4, (IV+d)
CB d E E	S E T	5, (IX+d)				CB d E E	S E T	5, (IV+d)
CB d F 6	S E T	6, (IX+d)				CB d F 6	S E T	6, (IV+d)
CB d F E	S E T	7, (IX+d)				CB d F E	S E T	7, (IV+d)
E 1	P O P	IX				E 1	P O P	IV
E 3	E X	(SP), IX				E 3	E X	(SP), IV
E 5	P U S H	IX				E 5	P U S H	IV
E 9	J P	(IX)				E 9	J P	(IV)
F 9	L D	SP, IX				F 9	L D	SP, IV

### 3. $\mu$ COM-82 ニーモニック $\longleftrightarrow$ 機械語 対照表

#### 8ビット・ロード

$\times$	I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX+d)	(IY+d)	(nn)	n
LD A, $\times$	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A nn	3E nn
LD B, $\times$			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 nn
LD C, $\times$			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E nn
LD D, $\times$			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 nn
LD E, $\times$			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		1E nn
LD H, $\times$			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		26 nn
LD L, $\times$			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		2E nn
LD (HL), $\times$			77	70	71	72	73	74	75							36 nn
LD (BC), $\times$			02													
LD (DE), $\times$			12													
LD (IX+d), $\times$			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 nn
LD (IY+d), $\times$			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 nn
LD (nn), $\times$			32 nn													
LD I, $\times$			ED 47													
LD R, $\times$			ED 4F													

#### 16ビット・ロード

$\times$	AF	BC	DE	HL	SP	IX	IY	nn	(nn)
LD AF, $\times$									
LD BC, $\times$								01 nn	ED 4B nn
LD DE, $\times$								11 nn	ED 5B nn
LD HL, $\times$								21 nn	2A nn
LD SP, $\times$				F9		DD F9	FD F9	31 nn	ED 6B nn
LD IX, $\times$								DD 21 nn	DD 2A nn
LD IY, $\times$								FD 21 nn	FD 2A nn
LD (nn), $\times$		ED 43 nn	ED 53 nn	22 nn	ED 73 nn	DD 22 nn	FD 22 nn		
PUSH $\times$	F5	C5	D5	E5		DD E5	FD E5		
POP $\times$	F1	C1	D1	E1		DD E1	FD E1		

#### ブロック転送

LDI	ED A0
LDIR	ED B0
LDD	ED A8
LD DR	ED B8

#### ブロック・サーチ

CP I	ED A1
CP IR	ED B1
CPD	ED A9
CP DR	ED B9

# 8ビット算術論理演算

×	A	B	C	D	E	H	L	(HL)	(IX +d)	(IY +d)	n
ADD A, ×	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
ADC A, ×	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUB ×	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SBC A, ×	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
AND ×	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
XOR ×	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
OR ×	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
CP ×	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INC ×	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DEC ×	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

# CPUコントロール

NOP	00
HALT	76
DI	F3
EI	FB
IM 0	ED 46
IM 1	ED 56
IM 2	ED 5E

# 16ビット算術演算

×	BC	DE	HL	SP	IX	IY
ADD HL, ×	09	19	29	39		
ADD IX, ×	DD 09	DD 19		DD 39	DD 29	
ADD IY, ×	FD 09	FD 19		FD 39		FD 29
ADC HL, ×	ED 4A	ED 5A	ED 6A	ED 7A		
SBC HL, ×	ED 42	ED 52	ED 62	ED 72		
INC ×	03	13	23	33	DD 23	FD 23
DEC ×	0B	1B	2B	3B	DD 2B	FD 2B

# エクステンジ

EX AF, AF'	08
EX DE, HL	EB
EX (SP), HL	E3
EX (SP), IX	DD E3
EX (SP), IY	FD E3
EXX	D9

# アキュムレータ操作

DAA	27
CPL	2F
NEG	ED 44
CCF	3F
SCF	37

ローテート, シフト

×	A	B	C	D	E	H	L	(HL)	(IX +d)	(IY +d)
RLC ×	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB 06	FD CB 06
RRC ×	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB 0E	FD CB 0E
RL ×	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB 16	FD CB 16
RR ×	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB 1E	FD CB 1E
SLA ×	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB 26	FD CB 26
SRA ×	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB 2E	FD CB 2E
SRL ×	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB 3E	FD CB 3E
RLD								ED 6F		
RRD								ED 67		

	A
RLCA	07
RRCA	0F
RLA	17
RRA	1F

ジャンプ, コール, リターン

×	UN COND	C	NC	Z	NZ	PE	PO	M	P	
JP ×, nn	C 3 n n	DA n n	D 2 n n	CA n n	C 2 n n	EA n n	E 2 n n	FA n n	F 2 n n	
JR ×, e	18 e-2	38 e-2	30 e-2	28 e-2	20 e-2					
JP (HL)	E 9									
JP (IX)	DD E 9									
JP (IY)	FD E 9									
CALL ×, nn	CD n n	DC n n	D 4 n n	CC n n	C 4 n n	EC n n	E 4 n n	FC n n	F 4 n n	
DJNZ e										10 e-2
RET ×	C 9	D 8	D 0	C 8	C 0	E 8	E 0	F 8	F 0	
RETI	ED 4D									
RETN	ED 45									

リスタート

RST 00H	C 7
RST 08H	C F
RST 10H	D 7
RST 18H	D F
RST 20H	E 7
RST 28H	E F
RST 30H	F 7
RST 38H	F F

ビット操作

×	A	B	C	D	E	H	L	(HL)	(IX +d)	(IY +d)
BIT 0, ×	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
BIT 1, ×	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
BIT 2, ×	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
BIT 3, ×	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
BIT 4, ×	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
BIT 5, ×	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
BIT 6, ×	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76	DD CB d 76	FD CB d 76
BIT 7, ×	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
RES 0, ×	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
RES 1, ×	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
RES 2, ×	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
RES 3, ×	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E
RES 4, ×	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
RES 5, ×	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
RES 6, ×	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
RES 7, ×	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
SET 0, ×	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
SET 1, ×	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
SET 2, ×	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6
SET 3, ×	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
SET 4, ×	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
SET 5, ×	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
SET 6, ×	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
SET 7, ×	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE

入 カ

IN A, n	DB n
IN A, (C)	ED 78
IN B, (C)	ED 40
IN C, (C)	ED 48
IN D, (C)	ED 50
IN E, (C)	ED 58
IN H, (C)	ED 60
IN L, (C)	ED 68
INI	ED A2
INIR	ED B2
IND	ED AA
INDR	ED BA

出 カ

OUT n, A	D3 n
OUT (C), A	ED 79
OUT (C), B	ED 41
OUT (C), C	ED 49
OUT (C), D	ED 51
OUT (C), E	ED 59
OUT (C), H	ED 61
OUT (C), L	ED 69
OUTI	ED A3
OTIR	ED B3
OUTD	ED AB
OTDR	ED BB



著 者 川村 清

監 修 FORESIGHT PLANNING SECTION

発行者 牧谷秀昭

発行所 秀和システムトレーディング株式会社

郵便番号107 東京都港区南青山2-3-21 小林ビル4 F

SHUWA SYSTEM TRADING CO., LTD.

KOBAYASHI BLDG,

2-3-21 MINAMIAOYAMA, MINATO-KU, TOKYO 107, JAPAN

印刷所 東京スガキ印刷株式会社

発行日 1982年11月1日

定価は表紙カバーに表示されています。







